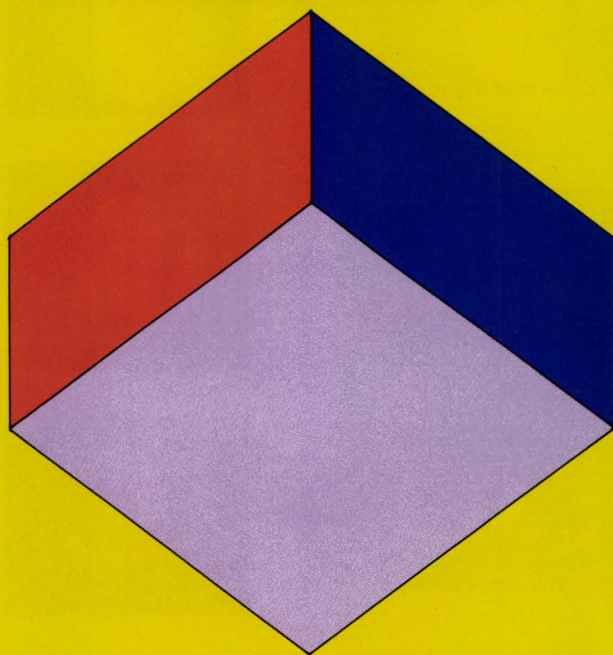
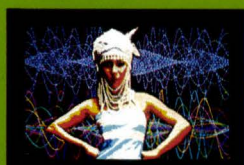
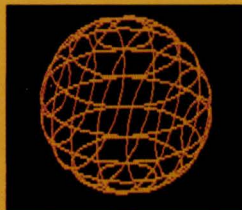
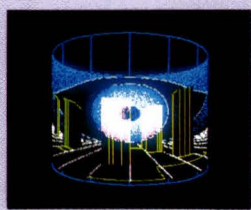
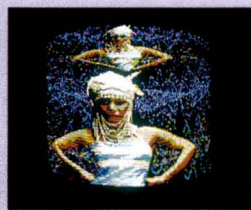
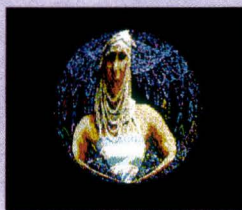
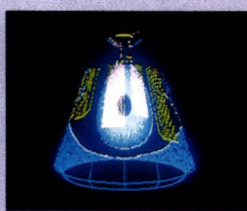
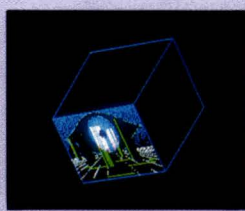
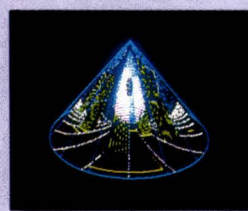
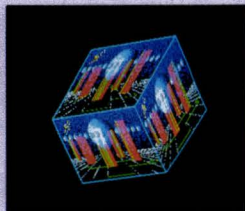
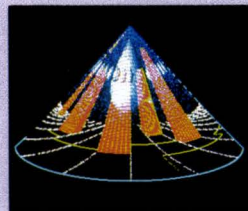
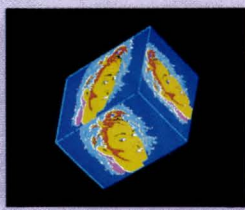
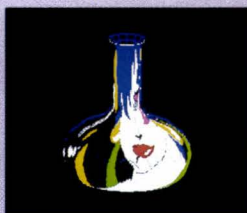
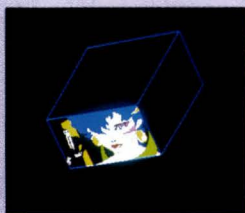
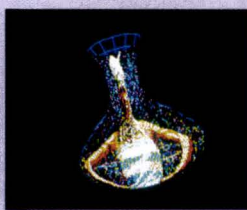
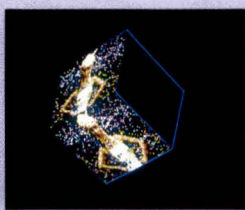
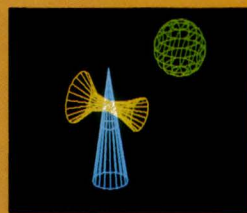
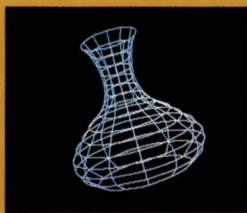
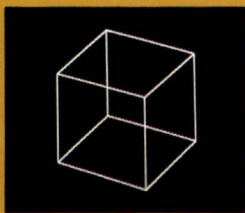
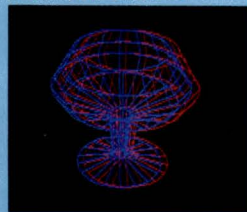
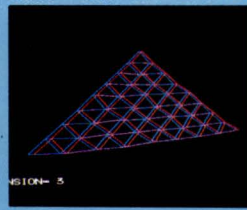
X1ターボ/X1シリーズで楽しむ

3次元グラフィックスの描き方

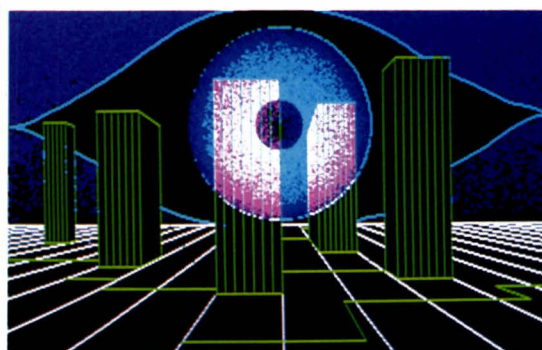
畠中兼司著

X1ターボ／X1シリーズで楽しむ

3次元グラフィックスの描き方

掲載プログラム・ガイダンス

3 D-STEREO
STEREO FRACTAL
RAY TRACING

ディスプレイをキャンバス代わりに

2次元デザインツール・プログラム(第2章)

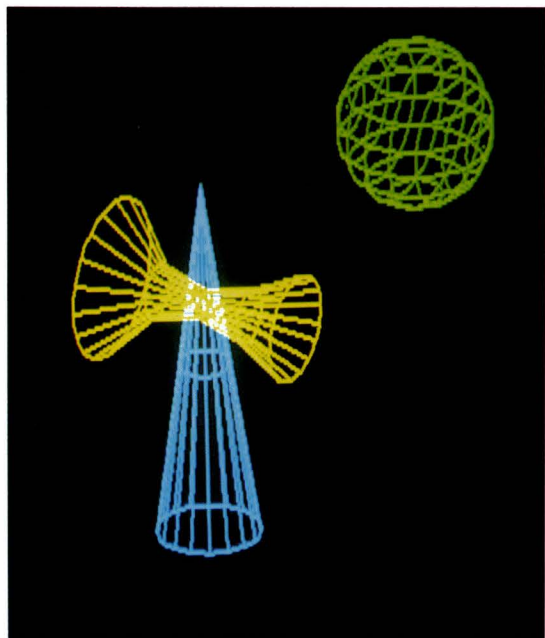
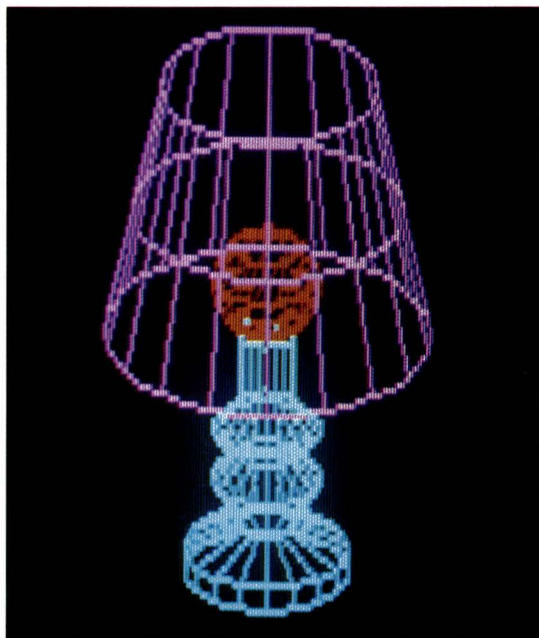
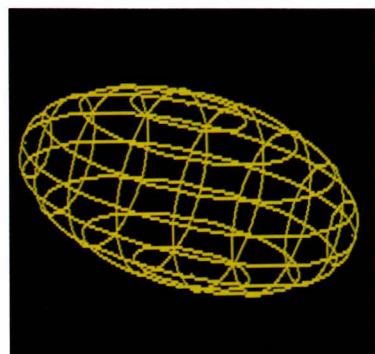
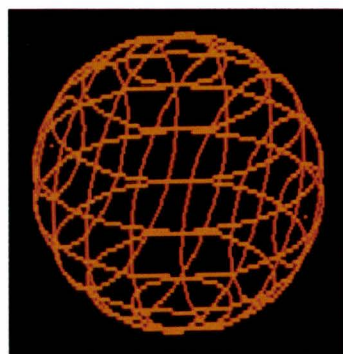
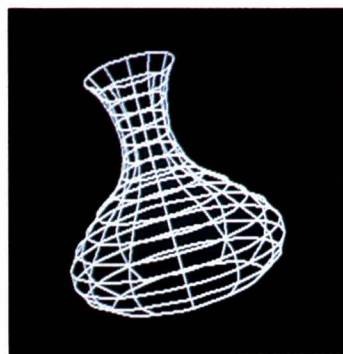
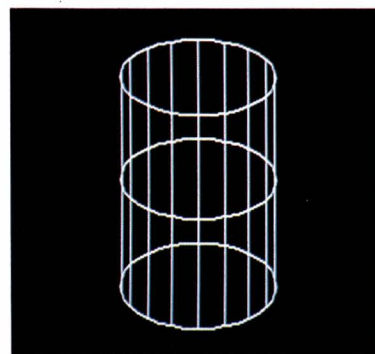
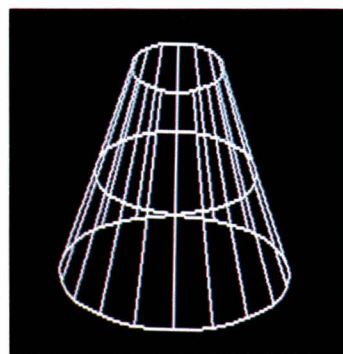
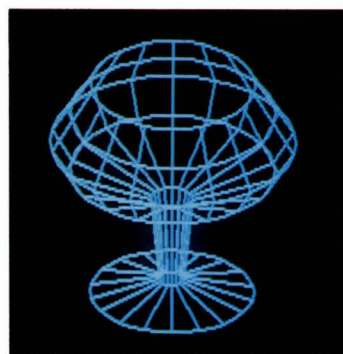
TECHNO WOMAN

COSMOS 1

COSMOS 2

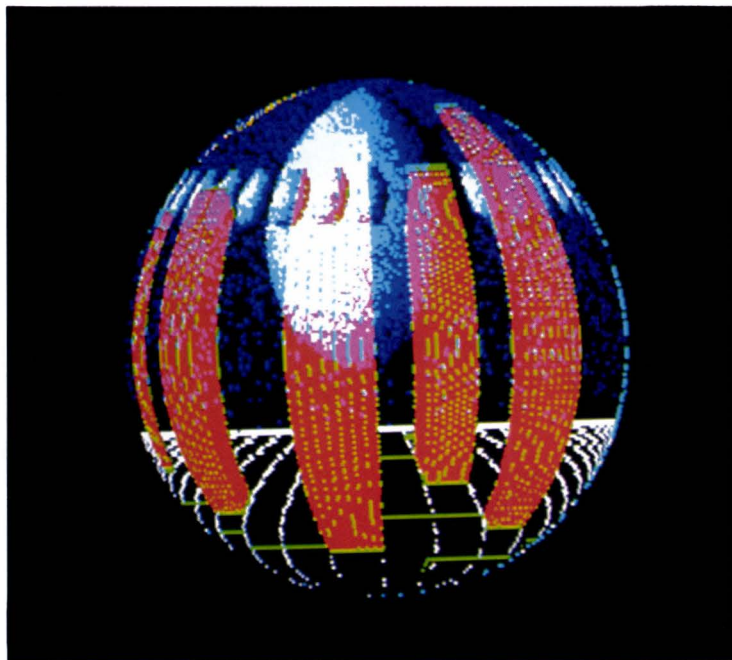
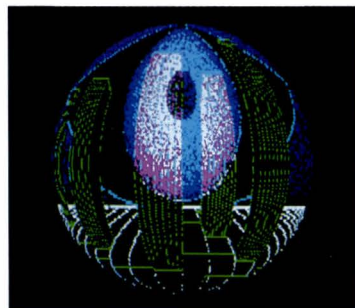
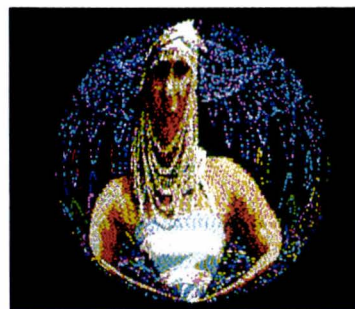
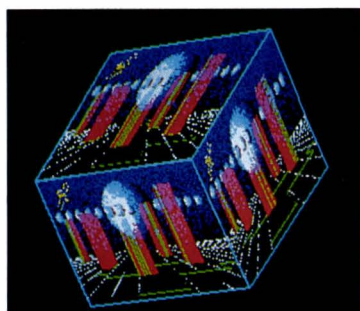
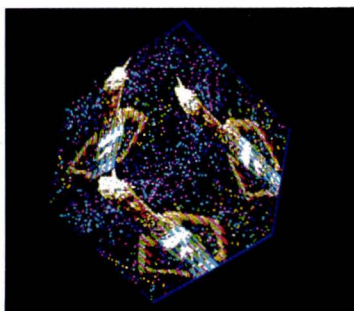
LADY

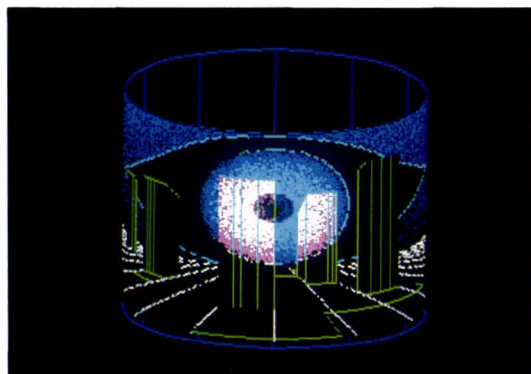
MAN AND WOMAN

ワイヤーフレームで立体図形を描く

3次元デザインツール・プログラム（第3章）

けん玉（剣：円錐，皿胴：回転体，玉：球体の合成）

ランプ（かさ：円錐，電球：球体，台：回転体の合成）

フットボール（X方向2倍の球体）

球体

回転体・つぼ

円柱

円錐

回転体・ワイングラス

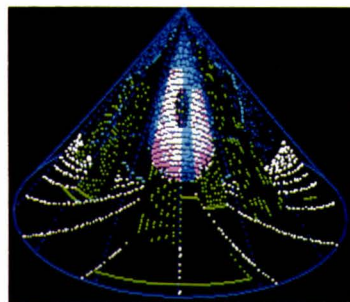
立体図形の表面に絵を描く

マッピング・プログラム(第4章)

COSMOS 1 + 球体

COSMOS 2 + 球体

TECHNO WOMAN + 球体

COSMOS 1 + 直方体

TECHNO WOMAN + 直方体

MAN AND WOMAN + 球体

COSMOS 2 + 円柱

MAN AND WOMAN + 円柱

TECHNO WOMAN ＋ 円柱

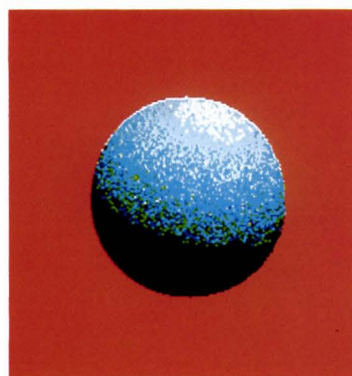
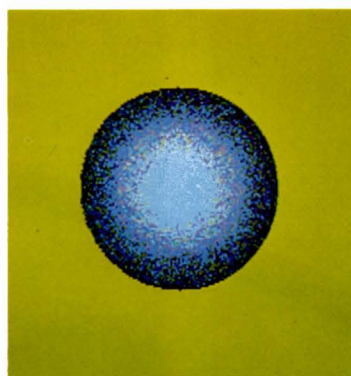
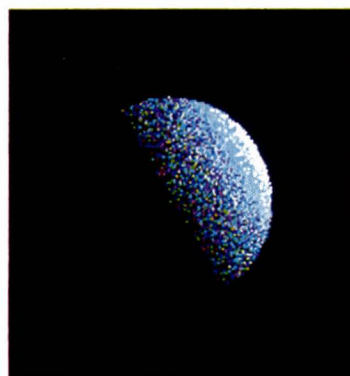
TECHNO WOMAN ＋ 円錐

COSMOS 1 ＋ 円錐

MAN AND WOMAN ＋ 回転体

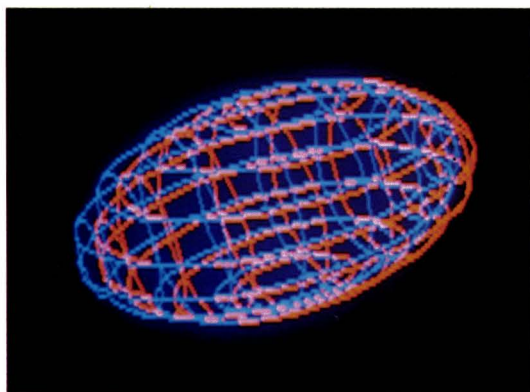
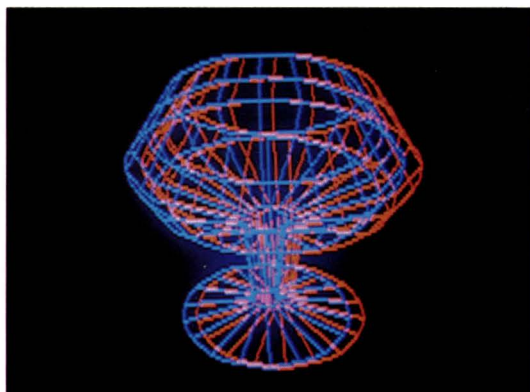
COSMOS 1 ＋ 回転体

MAN AND WOMAN ＋ 円錐

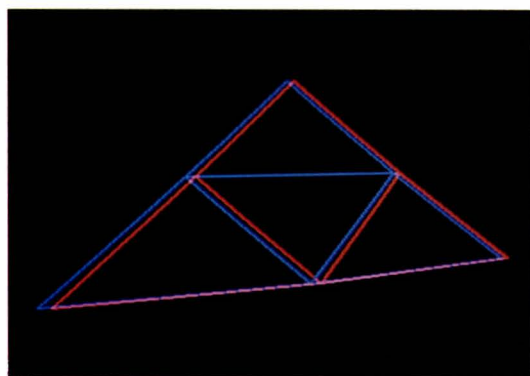
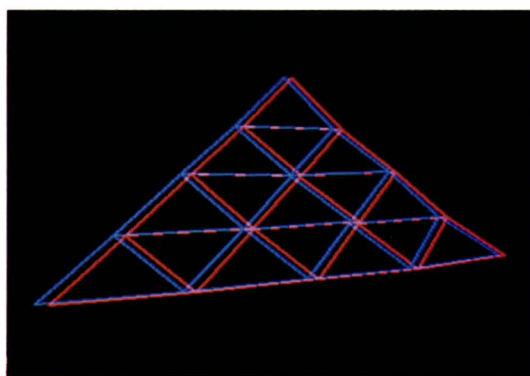
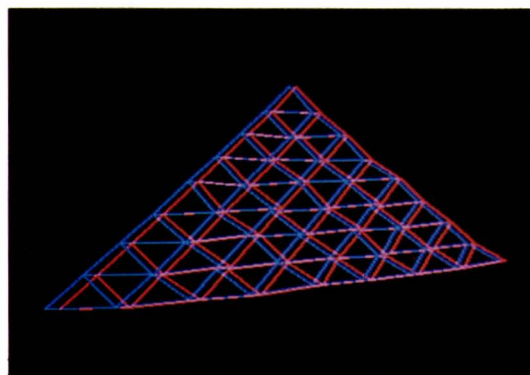
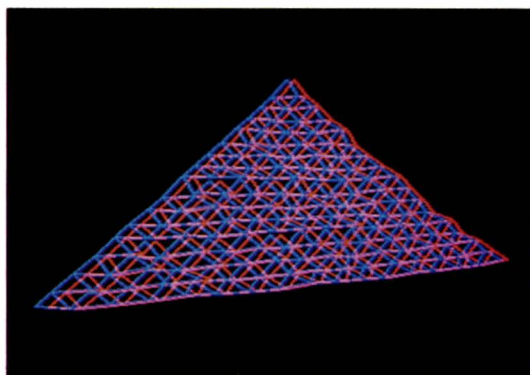
光の世界を演出する

レイトレーシング・プログラム(第5章)

EARTH 1

EARTH 2

EARTH 3

浮かび上がるワイヤーフレーム

ステレオグラフィックス・プログラム(第5章)

フットボール

ワイングラス

立体感で迫る山肌

ステレオフラクタル・プログラム(第5章)

1次のフラクタル

2次のフラクタル

3次のフラクタル

4次のフラクタル

まえがき

前著「パソコン・グラフィックスの作り方楽しみ方」を出版して約２年，読者の皆さまから数々の激励やご批評をいただきました。その間，Ｘ１シリーズも高密度・高性能化し，Ｘ１ターボシリーズへと発展し，まさしくパソコン実用化時代の先鞭をきるシリーズへと成長しました。メモリー容量も８ビット機では最高とも言えるまでに増設され，高級機種でしかできなかった，複雑なアプリケーション・ソフトウェアを組むことも可能となってきました。本書では，最高機種としてのハードウェアにふさわしいアプリケーションのいくつかを，紹介したいと思います。

３次元グラフィックスは，コンピュータで図形を表現するうえで非常に魅力ある方法といえます。その図形データの持ち方は，ハードウェアの記憶容量，計算スピード等の性能に負うところが大ですが，それを動かすためのソフトウェアしだいで，どんなものでも表現できるまでになりました。若干のスピードの遅さと煩わしささえ我慢すれば，何百万円，何千万円もするコンピュータにさえ負けずに動かせるようになります。

本書で紹介するソフトは，私たちが住んでいる３次元空間の物体を，パソコンで表現しようというものです。３次元物体をパソコンで表現するには，点と線で表す方法，面で表す方法，中身の詰まった物体（ソリッド）として表す方法などいろいろな手法があります。本書では，パソコンというハードウェアを考えて，よりスピーディに，よりリアルに表現するために，３次元を点と線，それに面で表現する方法について基礎から解説しています。皆さんにとっては，ディスプレイ画面に現れるグラフィックスの世界が，あまりにも自由に変化するため，自分のいる位置すら見失ってしまうかもしれません。焦らずに学んでください。

この３次元の世界は，まぎれもなく私たちが住んでいる世界なのです。それをコンピュータ内部で忠実に表現しているだけなのです。実在の空間と同じ空間をコンピュータ内部で作り上げる。実に楽しいことではありませんか。さあ，いっしょに３次元グラフィックスの世界へ飛び込んでみましょう。

本書は，さまざまな方々のご好意によって生まれました。特に，シャープ株式会社取締役総合デザイン本部長・坂下清氏，電子機器事業本部システム営業部長・奥時雄氏，同課長・樋口英男氏，細川俊吾氏，ソフト開発室部長・寳持（ほうじ）義昭氏には，数々のご教授ならびに機器のご提供でお世話をいただきました。わが仲間である総合デザイン本部の皆さん，そして電子機器事業本部Ｘ１関連の開発部門の皆さんには，大いなる激励をいただきました。グループ「彩夢」の諸君には，昼夜を問わずお世話をかけました。株式会社学習研究社の中山順一朗氏，秋山久義氏，小林幹彦氏，オフィスIN・OUTの長島道彦氏には，出版・編集の労をとっていただきました。皆さま方には，深く感謝いたします。

1985年12月　大阪にて

畠中　兼司

もくじ

●デザイン／仕事場201　●図版／日進工房＋海野アートプロ　●編集協力／オフィスIN·OUT　●写真／学研写真部

本書の構成と使い方

本書では，３次元グラフィックスを大いに楽しんでいただくために，代表的な手法を取り入れた10本のプログラムを用意しました。しかし，プログラムをただ実行するだけでは，皆さんが独自のグラフィックスを作り出すための発展は望めません。そこで，若干の煩わしさを感じられるかもしれませんが，３次元空間で図形を表現するうえで欠かせない，いくつかの基礎知識や基本のテクニックを，負担にならない程度にやさしく解説しています。さらに，２次元での画像処理を学んだうえで３次元処理のさまざまなテクニックをマスターするといったように，章を追って，段階的・系統的につながりを持たせた解説を試みています。ぜひ，初めから順に，じっくり取り組んでみてください。

以下に，本書の各章の内容を紹介しておきます。

第１章　３次元グラフィックスの基礎

コンピュータグラフィックス（CG）は，CAD／CAM，ビジネス，CAI，アニメーション，デザインというように，種々の分野で応用されています。それぞれの分野で表現される画像の形態（デザイン）は，大きく異なっています。例えば，CAD／CAM では，その製品の持つ形態の精密な寸法が重要であったり，また，ビジネスの分野では，表示されるグラフの目盛りや，その表現するシンボルが重要であったりします。しかし，どういった分野であっても，コンピュータによってそういった情報を処理するかぎりにおいては，コンピュータ内部にいかに効率よく絵をデータとして記憶させるかがカギであり，また，そのデータを基にして種々の図形操作を行うことが基本であるといえます。

第１章では，３次元グラフィックスの基礎として，主としてコンピュータ内部にどのように図形が作られていくのか，また，作られた図形がどのように変換されているのかなどについて，基礎から解説を行います。解説をわかりやすくするために，便利な行列式によって解説しました。行列は，現在の教育では，高校の数学で学ぶものですが，わかってしまうと便利なものです。学んでいない方，忘れてしまった方には，「付録３　マトリックス演算の基礎知識」で簡単に行列式の使い方について解説しています。そちらも参考にしながら，この章をよく理解してください。理屈などはどうでもいいから，とにかく描きたいという方は，この章をとばしていただいてもけっこうです。

第２章　２次元画像処理技術入門

前著「パソコン・グラフィックスの作り方楽しみ方」の中で紹介した「デザインツール」をさらにバージョンアップしたアプリケーション・ソフトを紹介しています。このツールを自在に活用することによって，ディスプレイ画面にさまざまな図形や絵を描くことができます。あなたのセンスを十分に発揮して，独創的な２次元グラフィックスを作り出してください。ここで描いた絵柄は，画像データをセーブしておくことによって，第４章で紹介するマッピングの原画として利用することができま

す。なお，以下に紹介するプログラムについても同様ですが，プログラムの組み立てを理解しやすくするため，変数表および流れ図（フローチャート）を紹介しています。

第3章　3次元図形処理技術入門

この章では，本格的な3次元図形を表現するためのデザインツール・プログラムを紹介しています。第1章でも，形状モデリングのさまざまな種類を紹介していますが，ここでは，ワイヤーフレーム・モデルまたはサーフェイス・モデルのデータ構造をとっています。実際に演算処理を行う場合には，第1章で解説した座標変換や透視変換の技術を核としてプログラミングする必要があります。ここで紹介しているプログラムには，そういったテクニックを実際に盛り込んでソフトウェアが構成されています。3次元の世界を理解するうえで，ぜひ，第1章と照らし合わせて読まれることをおすすめします。3次元空間に表される物体の姿を楽しんでください。

第4章　マッピング処理技術

マッピング技術は，CGの高等テクニックの1つとして注目すべきものです。

物体の表面のテクスチャを表す場合，表面パターンの微細な形状を，いちいち形状レベルのパッチで定義するのが実際的ではないことは，だれが考えても明らかなことです。このため，表面パターンは，2次元の画像処理技術で描き，物体形状は，3次元モデルで描くといったように分けてモデル化し，3次元形状モデルに2次元パターンをマッピングするといった手法をとります。

この方法は，一般にTexture mapping（テクスチャ・マッピング）と呼ばれ，2次元パターンを3次元モデルにはりつける技術として重要なものといえます。

第4章は，マッピング処理技術として，第2章で描かれた2次元画像処理図形を，3次元で表現されるプリミティブへはりつける技術（テクスチャ・マッピング）について，アプリケーション・ソフトウェアとともに解説し，紹介します。気長に一つ一つのグラフィックスの仕上がりのプロセスを楽しんでください。

第5章　図形処理技術の応用

第5章は，画像処理技術の応用として，レイトレーシング（光線追跡法），ステレオ・グラフィックス，ステレオ・フラクタルの各プログラムを紹介します。いずれも，最近のコンピュータ・グラフィックス・ブームの中で，最も高度で難しいテクニックばかりです。容易にわかりやすいように解説してありますが，より理解できるように，実際に動かしてみてください。ステレオ・グラフィックスに関しては，ステレオめがねをかけて鑑賞してください。立体像が浮かび上がります。

本書で使用するパソコンとBASIC

X1シリーズには，X1D系，X1F系そしてX1ターボにはmodel 10, 20, 30, 40に続いてターボIIと続々と機種がそろってきました。本書で採用したプログラムは，それらのすべての機種とBASICに対応できるように組んであります。ただ機種により若干の相違がありますので，アプリケーション・プログラムには，それぞれどの箇所をどのように変更すればいいのかを解説してあります。それに従って変更してご使用ください。

●X1ターボの場合

X1ターボのDISK・BASIC (CZ-8FB02) およびCASSETTE BASIC (CZ-8CB01) ご利用の読者は，このまま打ち込んでご使用ください。

ただし，model 10ご利用の読者は，別売グラフィックRAM (CZ-8BGR2)を使用すると，高解像度画面（640×400ドット）として使用できますが，標準機には実装されていませんので，低解像度画面（640×200ドット）のままで使用することになりますのでご注意ください。その場合は，画面表示の命令，つまりWIDTH命令はX1ターボ用，座標系はX1シリーズ用として打ち込んでください。

また，BASIC起動時に，[HELP]キーを押してNEW ON[CR]でご使用ください。

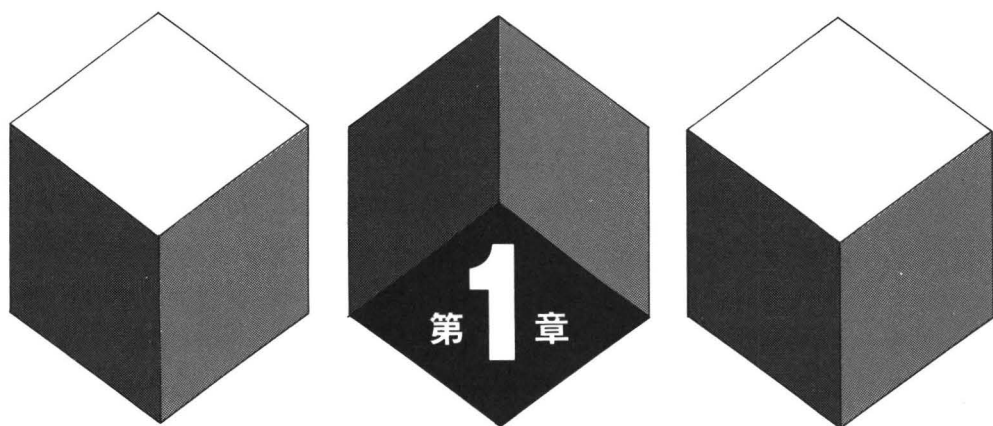
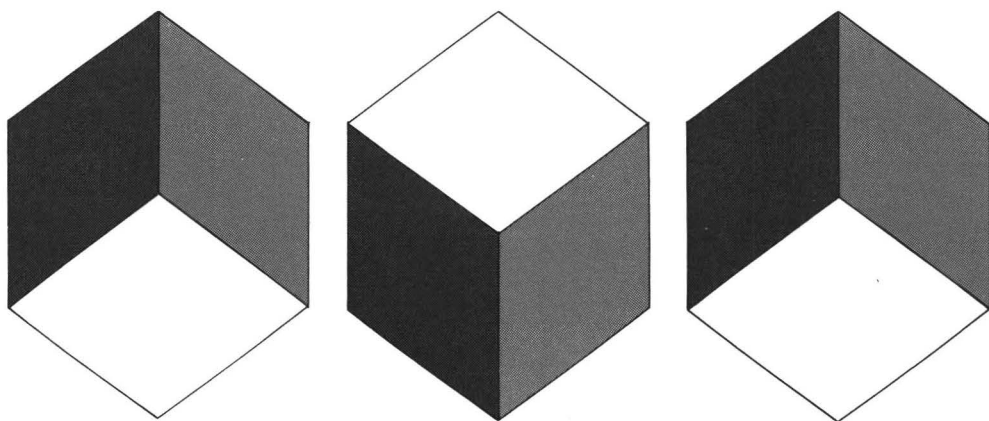
●X1シリーズの場合

X1シリーズのDISK BASIC (CZ-8FB01) およびCASSETTE BASIC (CZ-8CB01) ご利用の読者は，各プログラムに指示されたREM文，または変更点に従って打ち直してください。

ただメモリー容量の関係で，X1ターボシリーズより若干機能を低くおさえてある場合もありますので，ご了承ください。

●X1Fの場合

X1シリーズのNEW BASIC (CZ-8FB01 V.2.0, CZ-8CB01 V.2.0) ご利用の読者は，X1シリーズと同様にプログラムを変更してから入力してください。また，BASIC起動時に，NEW ON[CR]で使用してください。

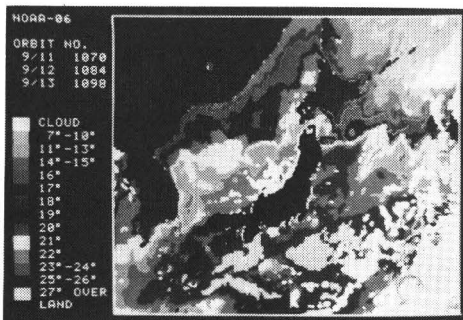
すべてのプログラムは，以上の変更内容で使用可能ですが，X1ターボ用 BASIC, NEW BASICには，NEWON命令で使用できるメモリー空間と使用命令の設定が可能となっています。詳細はそれぞれのBASICマニュアルを参照してください。

3次元グラフィックスの基礎

コンピュータ・グラフィックスの表現技術

コンピュータに図形を記述する方式として，大きく2通りの処理技術があります。

1つは，写真や絵をそのまま2次元のデータとしてメモリーに取り込んで処理するもので，**画像処理技術**と呼ばれているものです。これは，地球の気象状況や樹木の分布等，自然の実態を調査・解析するためのリモートセンシング技術や，脳の断面図や物体の材質処理といった，主として形状そのものが意味を持つ（X, Y, Z の情報を必要としていない）もので，写真のようなラスター情報そのものを処理する技術といえます。

2つ目の技術は，基本的に世の中の物体そのものの持っている大きさ等を記述するために，3次元のデータとして取り込んで処理するもので，**図形処理技術**と呼ばれているものです。これは，データ構造そのものが，X, Y, Z といった座標データを持っており，これらを処理することによってあらゆる形状表現を行うもので，データ構造のいかんによって，画像そのものの記述だけではなく，加工処理装置等を組み合わせて NC（数値制御）加工をするといったことが可能になります。

2つの方式は，相通じるものがあるわけですが，現在のところ，別々の研究分野で利用されており，両者の統合化が今後の大きな課題であるといえます。

画像処理技術	図形処理技術
写真のように，画像そのものが意味を持つもので，カメラやスキャナーなどを使ってピクセルのドット情報をメモリーに取り込み，それらを処理することによって表現する技術。	X，Y，Zの座標データを基本的に持ち，そのデータ構造のとり方によって，ワイヤーフレーム，サーフェイス，ソリッドなどのモデリングが可能な技術。
応用・リモートセンシング	応用・CAD/CAM，CAE，コンピュータアニメーション

●リモートセンシングの応用例――日本近海の温度分布（漁業情報サービスセンター）

●CAD／CAMの応用例――電卓の組立概念図（シャープ）

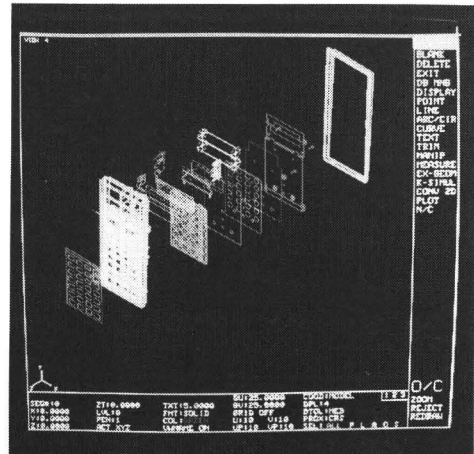



図1―1　コンピュータ・グラフィックス表現技術の分類

画像処理技術の基礎

通常，写真とか絵画といった画像を私たちの目に送りつける絵柄（カラー）には，X，Yの座標値はなく，1枚の原稿（あえてこう呼ぶ）上に画像情報（色や形状など）を蓄積して表示しています。この場合の原稿は，絵柄の情報を，三原色おのおのの連続的な濃度変化の組み合わせという形で記憶しているメモリー（記憶装置）と考えられます。人間の目は，この原稿に光を当てて，蓄積された情報を読み取っています。

では，これをコンピュータで処理するにはどうしたらいいのでしょう。

通常，コンピュータが扱える情報の形式は，ビットと呼ばれる2進数を組み合わせた電気信号で成り立っています。これに先ほどの画像情報を覚えこませるには，ビット形式に画像データを変換しなくてはなりません。これは電気的に行うわけですが，絵柄等の情報を2進数の電気信号に置き換える作業を，「デジタル化」と呼んでいます。

通常，絵柄をデジタル化するには，「スキャナー」を使用します。これは，複写機のようなものと考えていただければよく，あらかじめ用意された絵柄などの原稿を，スキャナーの原稿台に載せ，光のスポットをその原稿上に当てて走査させ，1点1点のデータをデジタル化していきます。カラー原稿の場合は，R（赤）G（緑）B（青）の光の三原色に分割して，フィルターを通して，それぞれの色に対応した光の量を受けて電気信号に変えていきます。

光を走査する場合のスポットの精度は，絵柄等の原稿をデジタル化するときの解像度に影響します。

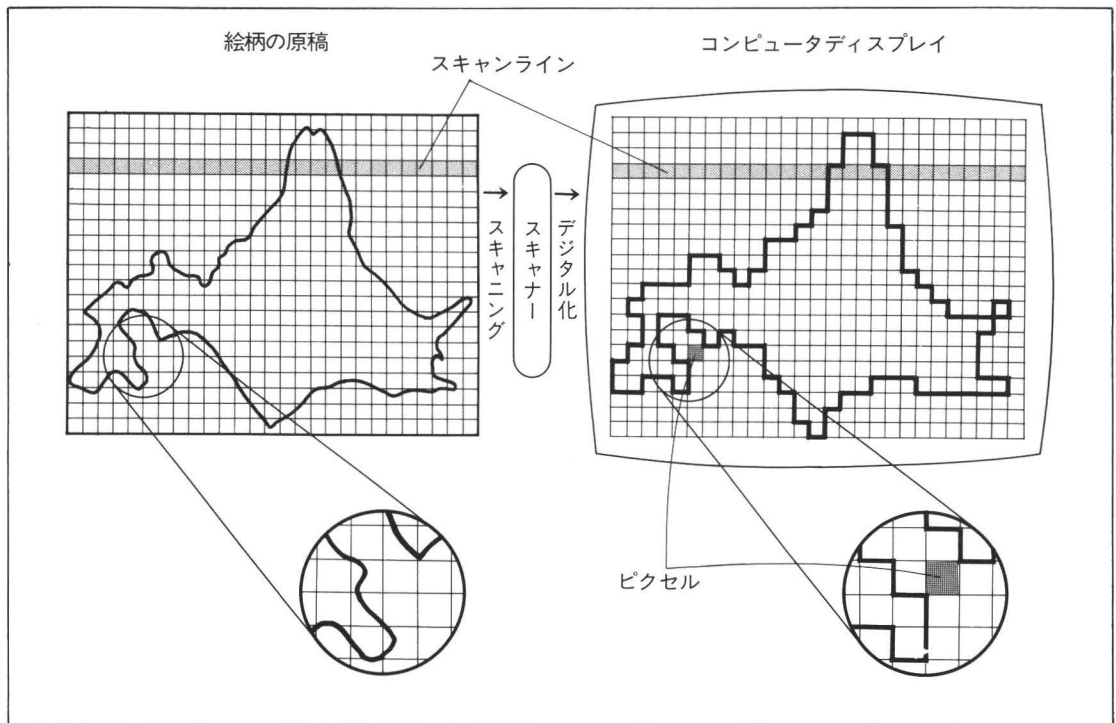



図１―２　画像処理の基本的な工程

よくミリ8本とか，ミリ12本とかいわれているのは，こういったスポットの精度を指していっているわけです。こういったスポットの精度は，ちょうど原稿を精度の単位である細かい方眼紙に分けたものと同様といえます。通常この方眼紙の1つのマス目を1ピクセルと呼んでいます。

私たちが今使おうとしているパソコンは，高解像度画面で640×400ドット＝256000ドット，つまり256000ピクセルの画面表示能力があります。例えば，32 cm×20 cm の原稿を用意している場合，画面全体に表示するには，ミリ2本の精度でいいわけです。通常はこういう荒いものではなく，ミリ20本くらいのものが産業用として使用されています。

デジタル化された情報は，基本的には，1ピクセルごとの色情報として記憶されます。記憶媒体としては，ディスプレイ画面のほか，フロッピーディスク，データレコーダ用カセットテープ等が考えられます。本書のプログラムは，X1ターボ/X1シリーズとも，ディスプレイ画面を複数枚持っているため，それらを使用して計算するほうが効率がいいという考え方から，記憶媒体として使用します（もちろん処理後の画像データは，フロッピーディスクおよびカセットテープにも記憶することができます）。

通常分割された微小な画素（ピクセル）は，1つ1つが三原色の輝度を持っています。多色の出る専門家用のディスプレイでは，R, G, B それぞれに8ビット，つまり256段階のメモリーを備えています（ルックアップテーブルという）。しかし，本書で扱うマイコンの場合は，R, G, B それぞれに1ビットしか持っていないため，8色しか表示できません。そのため，色情報を256階調持つために，ディザ法を用います。これには，組織的ディザ法とランダムディザ法の2種があります。このことに関しては，第5章のレイトレーシング（152ページ）で詳しく解説しています。

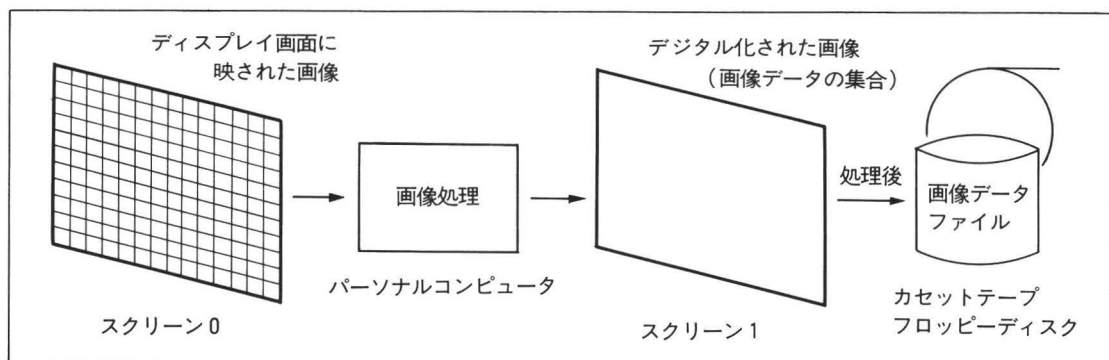



図1—3　画像のデジタル化とデータの保存

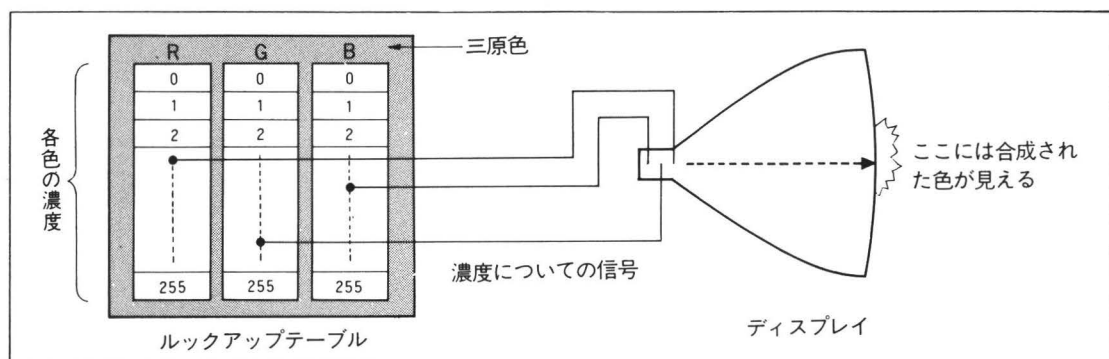



図1—4　三原色の濃度と合成される色

図形処理技術の基礎

世の中のすべての物体は，その物体の持っている大きさや色といったもので表すために，X，Y，Zという座標値を設定することができます。これらの値を用いて形状を記述するには，これらの座標値を設定するための仕組みを学ぶ必要があります。この仕組みを学ぶことにより，コンピュータ内部への図形の記述様式を理解することができます。

通常3次元（立体）の物体，例えば電気器具のテレビの外形形状は，大きく分けると，絵を表示するためのブラウン管，それを囲むベズルまわり，ボディ，チャンネル操作部などに分かれます。チャンネルのスイッチは，同一形状のものが複数個並んでいるし，ボディの形状，操作キーの形状は長方形の形をしているといえます。また，ボディ，チャンネル操作部に注目すると，さらに小さな形状の要素（エレメント）に分かれるのが容易に想像できます。こうしてみると一般的には，すべての物体は複数のエレメントから成り，また，階層構造をしているのがわかります。

ここでこれらの関係を整理してみると，ある階層のおのおののエレメントは，それが属する上位のエレメントに対し，ある変換によって関係づけられます。また，この変換の形式は，最上位に全体の座標系が存在し，階層構造を成す部分的な座標系群によって構成されることがわかります。これらのことから，コンピュータ内部に物体を形成するためには，種々の変換が必要なことがわかります。

コンピュータ内部にデータを形成する場合は，通常，図１—５でもわかるように，入力装置から，

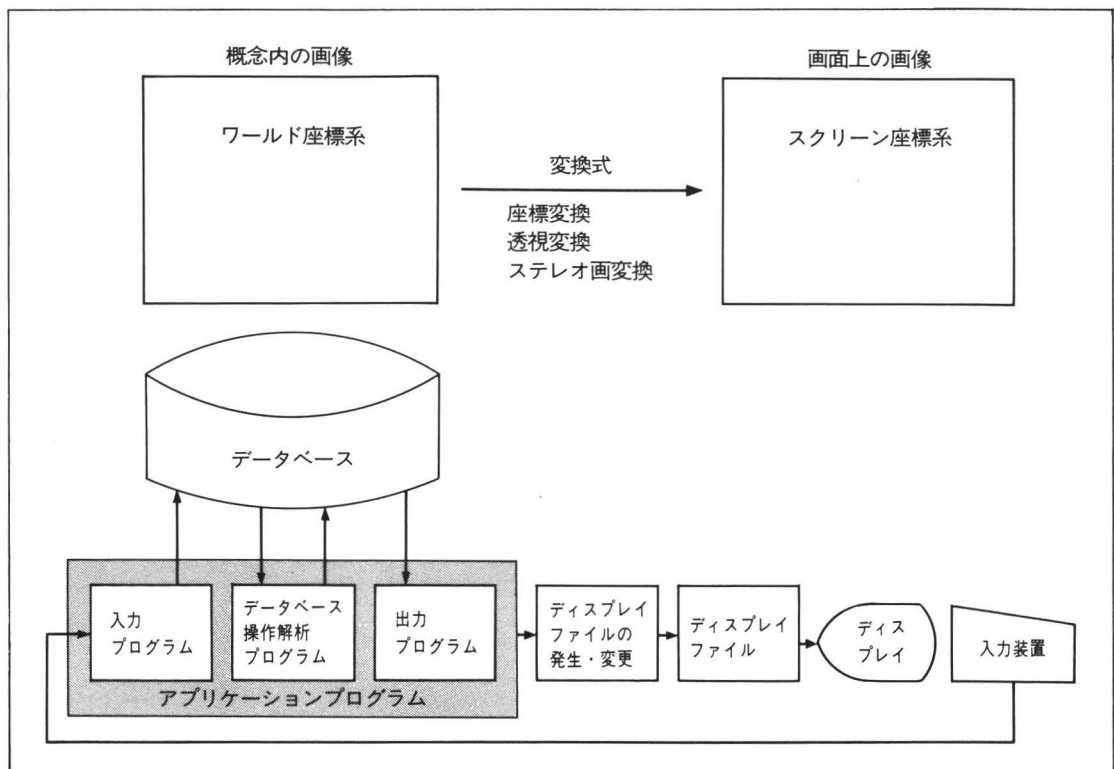



図1—5　グラフィックス・システムのソフトウェア関係図

その入力デバイスの座標系から，ある変換式をかけてワールド座標系に書き込んだ後，それを見せるための出力装置であるディスプレイ装置の特定な座標系（スクリーン座標系）に変換させて出力します。

ここで変換のためのテクニックが必要となるわけですが，大きくワールド座標系に描かれた図形を，移動，回転，拡大，縮小等を行うための座標変換，透視図的に見せる透視変換，物体を立体的に見せるステレオグラフィックス変換（基本的には座標変換，透視変換の組み合わせ）等のテクニックが必要になります。それぞれについて解説します。

形状モデリングの種類

図形処理技術の骨格を成す形状モデリングの表現技術には，現在使われているモデルとして3つの種類があります。

①ワイヤーフレーム・モデル

②サーフェイス・モデル

③ソリッド・モデル

それぞれ次のような特徴があります。

ワイヤーフレーム・モデル

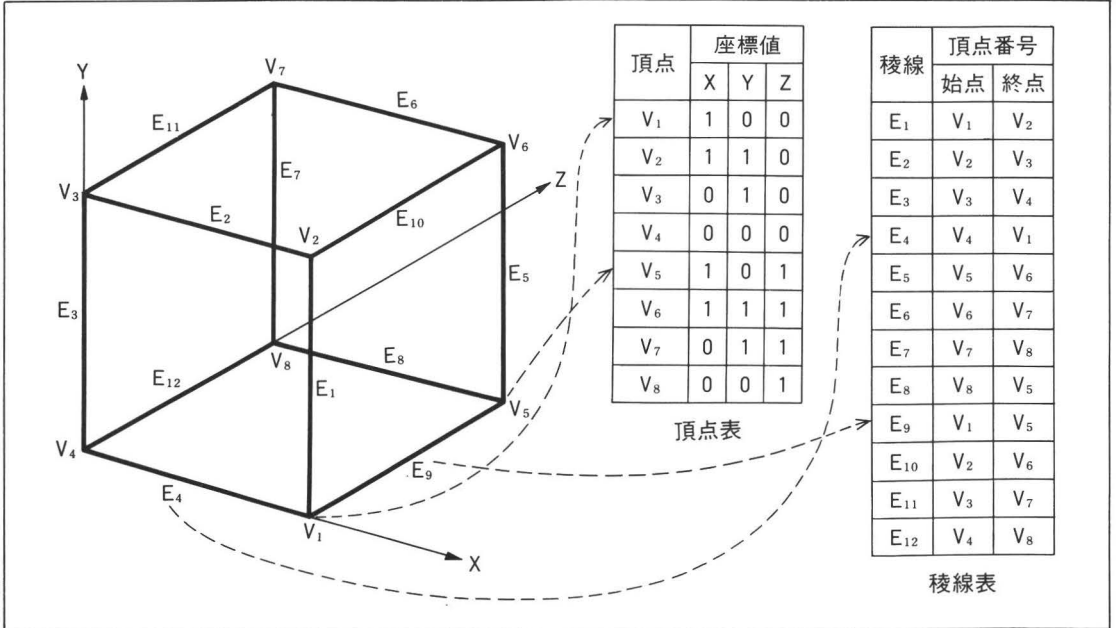
コンピュータに図形を記述させようとした場合に，どんなすばらしいシステムでも，取り扱うデータやプログラムに不備な点があれば，無用の長物となってしまいます。コンピュータ内に形状を表現する要素中でいちばんシンプルなものに点があります。しかし，この点だけでは，形を表現することは不可能に近いこととなります。点をたくさん並べて，つまり線にして初めて形状が記述できることになります。線さえあればどんな形状も，一応記述することは簡単になります。

ワイヤーフレーム・モデルとは，読んで字のごとく，まさにワイヤー（線）のフレーム（骨組）で

頂点	座標値		
	X	Y	Z
V_1	1	0	0
V_2	1	1	0
V_3	0	1	0
V_4	0	0	0
V_5	1	0	1
V_6	1	1	1
V_7	0	1	1
V_8	0	0	1

頂点表

稜線	頂点番号	
	始点	終点
E_1	V_1	V_2
E_2	V_2	V_3
E_3	V_3	V_4
E_4	V_4	V_1
E_5	V_5	V_6
E_6	V_6	V_7
E_7	V_7	V_8
E_8	V_8	V_5
E_9	V_1	V_5
E_{10}	V_2	V_6
E_{11}	V_3	V_7
E_{12}	V_4	V_8

稜線表

図Ｉ－6　ワイヤーフレーム・モデルのデータ構造

描かれるデータ構造を持つ形状モデリングを指しています。つまり，コンピュータ内部のデータ構造としては，点とそれらを結ぶ線のみで記述するものを指しています。そのため，非常にシンプルなデータ構造でメモリーも少なくてすみますが，円柱や円錐などの稜線以外に輪郭線がないと表現しにくい形状に関しては，ワイヤーフレームで記述を行うことは難しくなります。

また，ワイヤーフレームは線のみの形状であり，面情報以上のデータを持っていないことから，面構成の範囲とか，隠線消去，断面図表示，体積計算等といったことができない反面，データ構造の単純さから計算のスピード化や記憶容量の簡易さといった長所も見受けられます。

サーフェイス・モデル

サーフェイス・モデルは，ワイヤーフレーム・モデルのデータ構造に面データを追加したものです。面データを追加することから，ワイヤーフレーム・モデルで不可能とされていた隠線消去や，断面図表示，相貫線表示，表面積計算といった計算が可能となります。また，コンピュータ・グラフィックス（CG）にとって非常に大切な3次元モデルのリアルな画像表示であるレイトレーシングや2次元画像の3次元図形の面上へのマッピングなどが可能になってきます。

しかし面データは持っているものの，面と面との間には構造上の関係値を持っていないために，例えば，物体が面のどちら側に属しているかといったようなことは計算不可能です。また，サーフェイス（表面）のデータ構造しか持っていないため，物体内部の計算である重量や重心といったマス・プロパティの計算は不可能です。しかし，その持っている簡易さはすばらしいものがあり，現在開発，実用化されているCAD/CAMシステムのほとんどが，サーフェイス・モデルのデータ構造により記述されています。

ソリッド・モデル

ソリッド・モデルは，中身の詰まったソリッドな（固形の）立体を表現するモデルを指していいます。このモデルは，現在は基本的に研究途上の表現技術で，その表現方法には大きく3つの方法が

面	稜　番　号			
F_1	E_1	E_2	E_3	E_4
F_2	E_5	E_6	E_7	E_8
F_3	E_1	E_9	E_5	E_{10}
F_4	E_2	E_{10}	E_6	E_{11}
F_5	E_{11}	E_3	E_{12}	E_7
F_6	E_{12}	E_4	E_9	E_8

面表

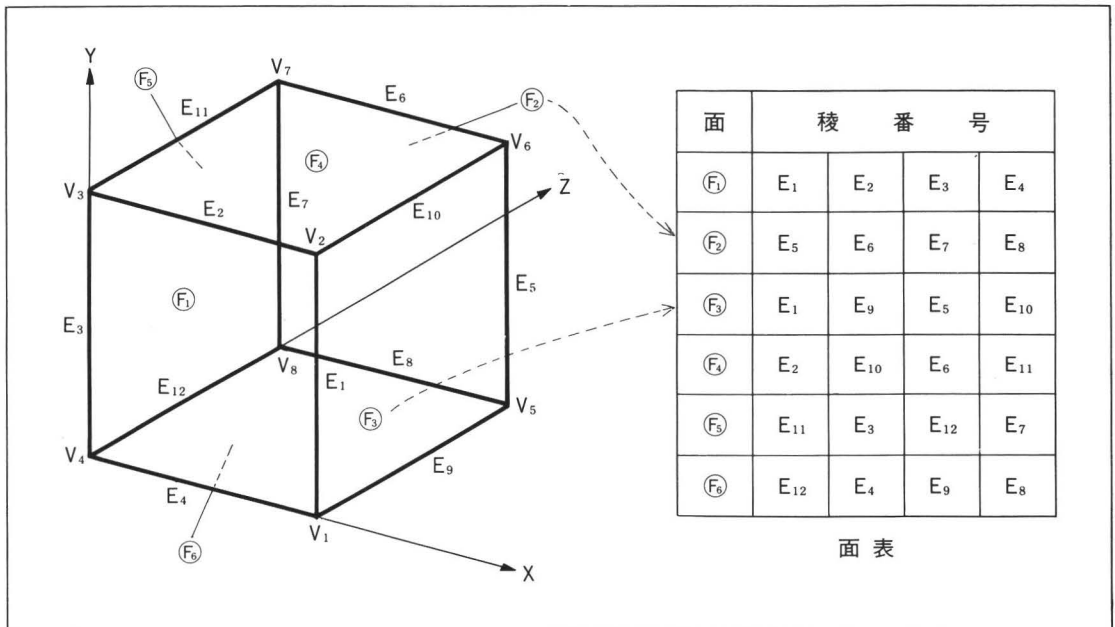



図1―7　サーフェイス・モデルのデータ構造

あります。

①B-reps 表現（Boundary representation）

②CSG 表現（Constructive Solid Geometry）

③Voxel 表現

1つ目の B-reps 表現とは，基本的にはサーフェイス・モデルの延長線上に位置するもので，面のどちら側に物体が存在しているのかを表現するために，ある面を物体の外側から見て，その面を構成する稜線を左回りに並べることにより，面の方向が物体の外側に向いていると定義し，面の外側と内側の判定をしているものです。これにより，物体の中身の情報，つまり立体情報を持つことになり，重心や断面形状といったマス・プロパティの計算が容易となります。

3次元物体を表現する場合，基本的には，物体とそれを囲む面，面とそれを囲む稜線，稜線とそれを形成する2個の頂点というように，「立体→面→稜線→頂点」というツリー構造（木構造）で記述します。その立体データを，ある面を物体の外側から見て，その面を構成する稜線を左回りに並べることにより，面の方向が物体の外側を向いていると定義すれば，面の外側と内側の判別が可能となり，ソリッド・モデルのデータ構造となります。図1－7のサーフェイス・モデルの面データを B-reps 表現によって並べ換えたものが図1－8です。

このように，3次元物体の形状をその境界面で表現する方法を境界表現（Boundary representation）と呼び，ワイヤーフレーム・モデル，サーフェイス・モデルの延長線上にあるテクニックとして，ほとんどの CAD/CAM システムが採用を図っています。

図1－9のように，3次元物体の1つの稜線をはさんで左右に2個の面があり，その形が鳥が翼を広げたような形状をしている点から，ウィングド・エッジと呼ばれています。この方式は，サーフェイス・モデルと共通の考え方ができるというほかは，直方体のような簡単なものとはいかず，入力が困難なほか，データ構造も複雑で，かつ大容量のメモリーを要するという欠点を持っています。

面	稜	番	号	
F1	E3	E4	E1	E2
F2	E6	E5	E8	E7
F3	E10	E1	E9	E5
F4	E11	E7	E12	E3
F5	E2	E10	E6	E11
F6	E12	E8	E9	E4

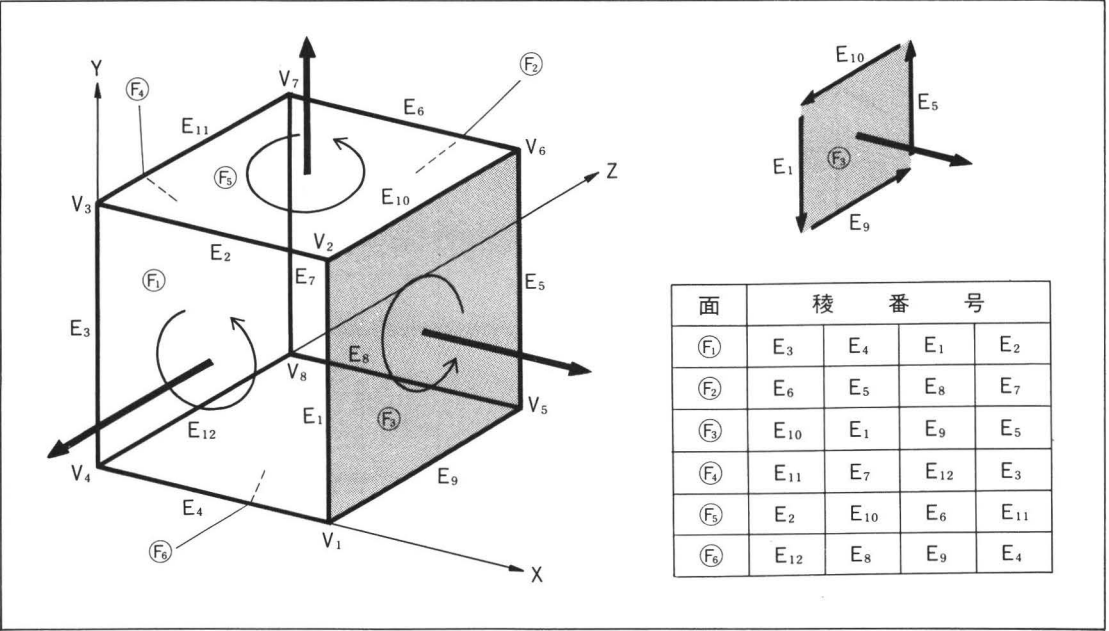



図1－8　B-reps表現によるソリッド・モデルのデータ構造

2つ目の CSG 表現（Constructive Solid Geometry）とは，形状記述を従来のように，「頂点→稜線→面→立体」に置き換えていくのではなく，初等幾何学で表現します。球体や円筒，円錐といったプリミティブ（基本形状）に置き換えて，それらを集合演算する組み合わせによって形状記述する方法です。

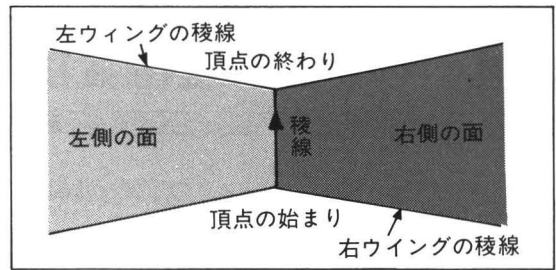



図1—9　ウィングド・エッジのデータ構造

定義した過程そのままの状態でデータ構造に記憶しているために，データ構造そのものがコンパクトであり，データの作成方法，修正等も容易です。しかし，従来型の設計手法では形状記述ができにくいことや，図面の作成や展開図の作成，表面積計算が困難といった欠点があります。

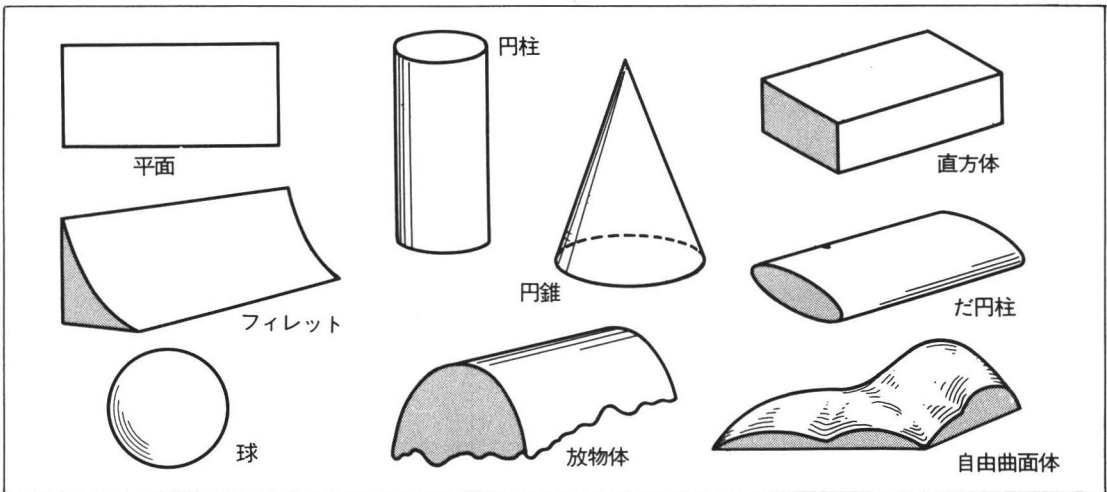



図1—10　プリミティブの例(北大TIPS-1による)

最後の Voxel 表現とは，3次元空間を小さな単位立方体に分割し，物体がこの単位立方体を占めるかどうかによって0，1の情報で表現する方法です。3つの方法のうち，最もシンプルな構造といえます。しかし，実際に実用化しうる精度を得るために単位立方体に分割するには，相当量のデータが必要となるので，ソフトウェアで処理するには，膨大な時間を要することとなります。

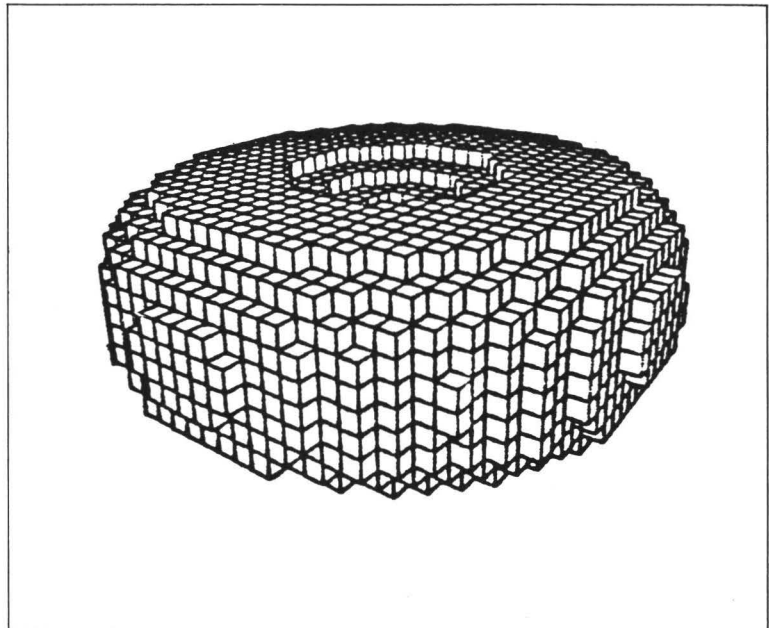



図1—11　Voxel表現の例

座標変換技術の基礎

コンピュータ・グラフィックスの重要な基礎技術の1つに座標変換技術があります。

説明をわかりやすくするために，まず3次元座標変換の基礎となる2次元の変換技術について話を進めます。次いでそれらのベースに立って，3次元の変換技術の解説をします。なお，変換技術を解説するために必要な数学的知識は，変換式を示してなるべく詳細に解説していますので，おわかりいただけると思います。また，座標変換に関係しているマトリックス演算を取り上げていますが，わからない読者は，「付録2　マトリックス演算の基礎知識」（178ページ）を参考にしてください。

2次元の座標変換

座標変換で最も基本的な変換には，次の3つがあります。

(A) 移動（Movement）

(B) 拡大・縮小（Scaling）

(C) 回転（Rotation）

移動（Movement）

座標軸 X, Y 上の1点 P(x, y) が，P′(x′, y′) に移動する場合には，移動量を t_x, t_y とすると次の関係式に表すことができます。

$$\begin{cases} x' = x + t_x \\ y' = y + t_y \end{cases} \qquad ①$$

これをあらためてマトリックス演算のための行列式で書き直すと，次のように書き表すことができます。

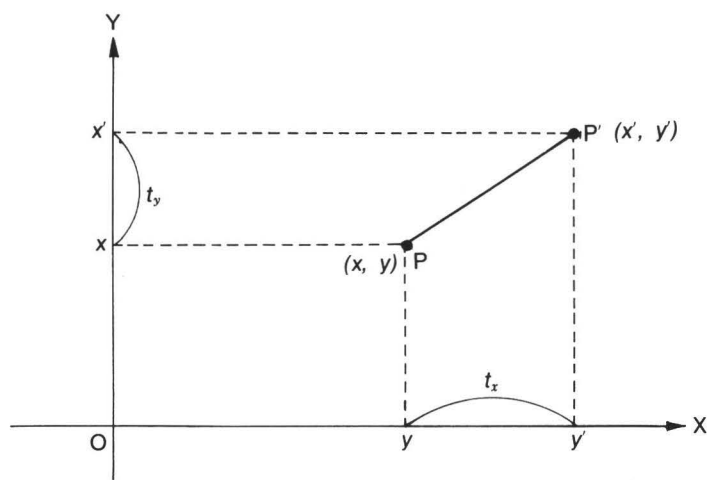



図1―12　点Pの移動

$$[x' \quad y'] = [x \quad y]\begin{bmatrix} 1 & 1 \\ t_x & t_y \end{bmatrix} \quad \text{②}$$

これは，ベクトル加算の形式で書き表していますが，通常，図形変換の場合は，１回の変換式で完了してしまうということはなく，いろいろな複数の変換式を組み合わせて利用することがあります。そのために，すべての変換処理を同一のものとしておく必要があります。そこで，同時座標系(Homogeneous coordinate system)の概念を導入する必要があります。

同時座標系では，座標を表す次元を１つ増やして，②式を変更して次のように書き表すことができます。

$$[x' \quad y' \quad 1] = [x \quad y \quad 1]\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ t_x & t_y & 1 \end{bmatrix} = (x + t_x \cdot y + t_y \cdot 1) \quad \text{③}$$

拡大・縮小(Scaling)

原点（0，0）を中心として拡大・縮小する場合は，点(x, y)がx軸方向にS_x倍，y軸方向にS_y倍されます。拡大・縮小後のそれぞれの座標を(x', y')とすると，次のように表すことができます。

$$\begin{cases} x' = S_x \cdot x \\ y' = S_y \cdot y \end{cases} \quad \text{④}$$

（ただし，$S_x > 1$, $S_y > 1$のときは拡大，$0 < S_x < 1$，$0 < S_y < 1$のときは縮小）

通常の場合は，$S_x = S_y$となり，$S_x \neq S_y$の場合は図形自体が変形し，拡大・縮小とは異なります。

これを行列式で書き直すと，次のようになります。

$$[x' \quad y'] = [x \quad y]\begin{bmatrix} S_x & 0 \\ 0 & S_y \end{bmatrix} \quad \text{⑤}$$

同時座標系であらためて書き換えると，次のようになります。

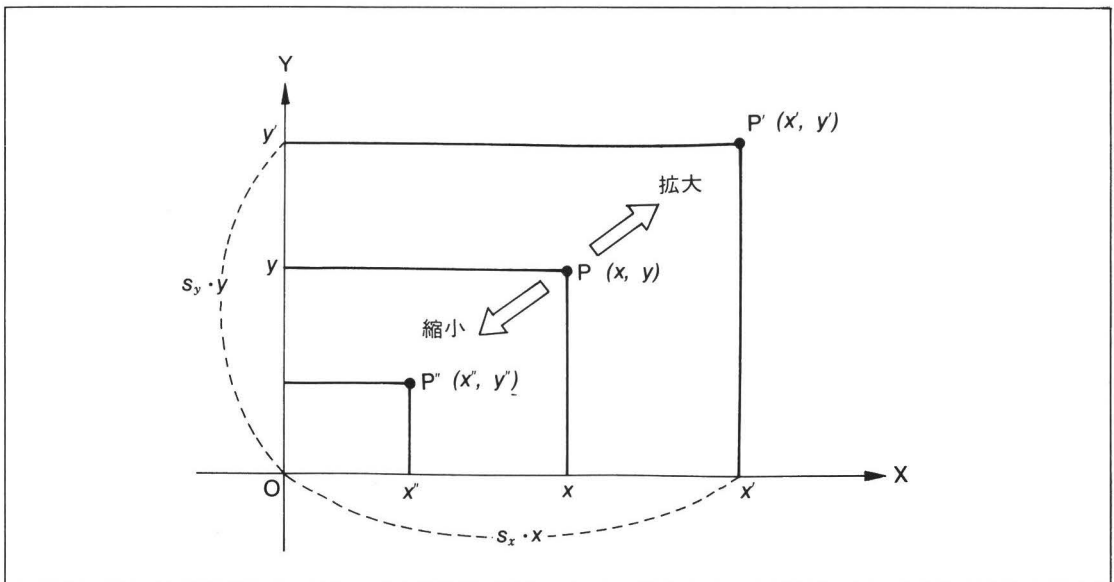



図１―13　点Pの拡大・縮小

$$[x' \quad y' \quad 1]=[x \quad y \quad 1]\begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad ⑥$$

回転(Rotation)

原点を中心として時計まわりに角度 θ だけ回転させる場合には，初期座標を (x, y)，回転後の座標を (x', y') とすると，極座標形式を利用してそれぞれ次のようになります。

$$\left.\begin{array}{l} x=r\cos\alpha \\ y=r\sin\alpha \end{array}\right\} \quad ⑦$$

$$\left.\begin{array}{l} x'=r\cos(\alpha-\theta)=r(\cos\alpha\cos\theta+\sin\alpha\sin\theta) \\ y'=r\sin(\alpha-\theta)=r(\sin\alpha\cos\theta-\cos\alpha\sin\theta) \end{array}\right\} \quad ⑧$$

⑦，⑧式より，次のようになります。

$$\left.\begin{array}{l} x'=x\cos\theta+y\sin\theta \\ y'=-x\sin\theta+y\cos\theta \end{array}\right\} \quad ⑨$$

これを行列式で書き表すと，次のようになります。

$$[x' \quad y']=[x \quad y]\begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \quad ⑩$$

同時座標系で書き表すと，次の式が得られます。

$$[x' \quad y' \quad 1]=[x \quad y \quad 1]\begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad ⑪$$

また，任意点 (R_x, R_y) を中心とした回転（時計方向に角度 θ）は，初期座標を (x, y)，回転後の座標を (x', y') とすると，③式と⑪式を組み合わせて，まず回転の中心 (R_x, R_y) を原点に移動し回転を行った後に元の位置に移動すればよいわけです。そこでまず，③式において移動量が，

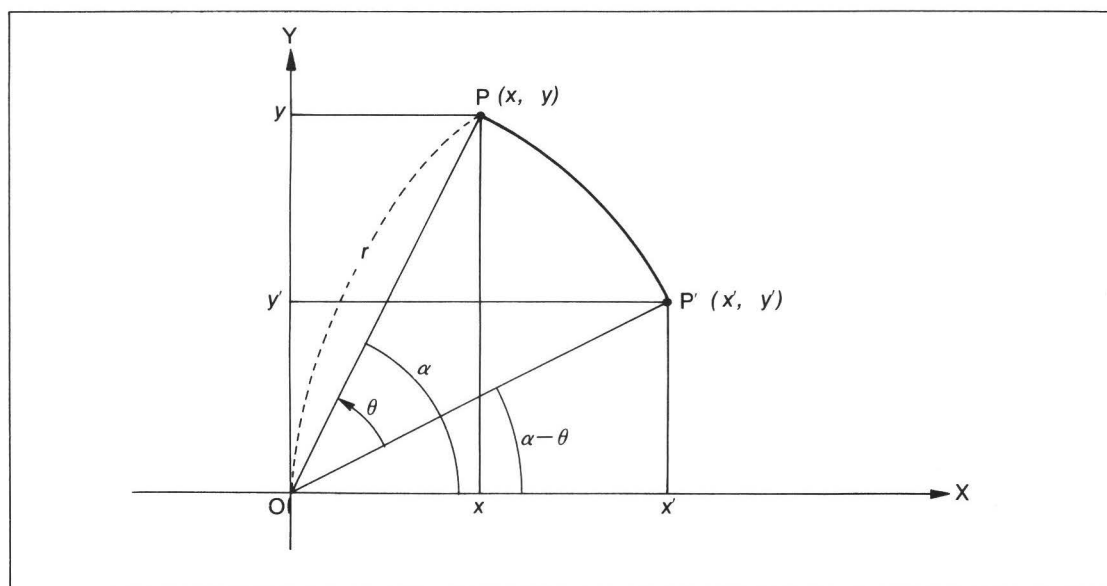



図1—14　点Pの回転

$T_x = -R_x$

$T_y = -R_y$

となり，移動後の座標を（x_1, y_1）とすると，次の式になります。

$$[x \quad y \quad 1]\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -R_x & -R_y & 1 \end{bmatrix} = [x_1 \quad y_1 \quad 1] \quad ⑫$$

次に，角度 θ だけ回転した座標を（x_2, y_2）とすると，次のように書き表せます。

$$[x_1 \quad y_1 \quad 1]\begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} = [x_2 \quad y_2 \quad 1] \quad ⑬$$

この状態では，回転の中心点（R_x, R_y）が原点に移動しているので，（x', y'）を求めるためにもとに戻すには，⑫式の逆マトリックスをとればよいので，次のように表現できます。

$$[x_2 \quad y_2 \quad 1]\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ R_x & R_y & 1 \end{bmatrix} = [x' \quad y' \quad 1] \quad ⑭$$

⑫⑬⑭式より，次の式が得られます。

$$[x' \quad y' \quad 1] = [x \quad y \quad 1]\begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ a & b & 1 \end{bmatrix} \quad ⑮$$

ただし　$a = -R_x\cos\theta - R_y\sin\theta + R_x$

$b = R_x\sin\theta - R_y\cos\theta + R_y$

3次元の座標変換

2次元での座標変換について解説しましたが，3次元の場合も同様に考えられます。

移動(Movement)

3次元座標系での1点 P（x, y, z）が，P′（x', y', z'）に，それぞれX軸，Y軸，Z軸方向の移動量 T_x, T_y, T_z だけ移動したとすると，そのマトリックス演算の行列式は，同時座標系では，次のように表現できます。

$$[x' \quad y' \quad z' \quad 1] = [x \quad y \quad 1]\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ T_x & T_y & T_z & 1 \end{bmatrix} = [x+T_x \quad y+T_y \quad z+T_z \quad 1] \quad ⑯$$

拡大・縮小(Scaling)

3次元座標系での1点 P（x, y, z）がX軸，Y軸，Z軸方向に，それぞれ S_x, S_y, S_z 倍された点を P′（x', y', z'）とすると，次の式で表すことができます。

$$[x' \quad y' \quad z' \quad 1]=[x \quad y \quad z \quad 1]\begin{bmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}=[S_x \cdot x \quad S_y \cdot y \quad S_z \cdot z \quad 1] \text{——} ⑰$$

回転(Rotation)

3次元座標系での軸は，右手系と左手系の2通りが考えられますが，本書では，回転の方向は各軸から原点に向かって，反時計方向を正とする左手系を基準として採用することにします。図形変換は，X軸，Y軸，Z軸それぞれに関する回転が考えられます。

●X軸中心とした回転

X軸のまわりに角度θだけ時計方向に回転させるには，回転の中心となるX軸の座標は変化しないので，⑪式を3次元へ拡張すればいいのです。それを式で表すと次のようになります。

$$[x' \quad y' \quad z' \quad 1]=[x \quad y \quad z \quad 1]\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & \sin\theta & 0 \\ 0 & -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \text{——} ⑱$$

●Y軸を中心とした回転

同様に，次のように書くことができます。

$$[x' \quad y' \quad z' \quad 1]=[x \quad y \quad z \quad 1]\begin{bmatrix} \cos\theta & 0 & -\sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ \sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \text{——} ⑲$$

●Z軸を中心とした回転

同様に，次の式が得られます。

$$[x' \quad y' \quad z' \quad 1]=[x \quad y \quad z \quad 1]\begin{bmatrix} \cos\theta & \sin\theta & 0 & 0 \\ -\sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \text{——} ⑳$$

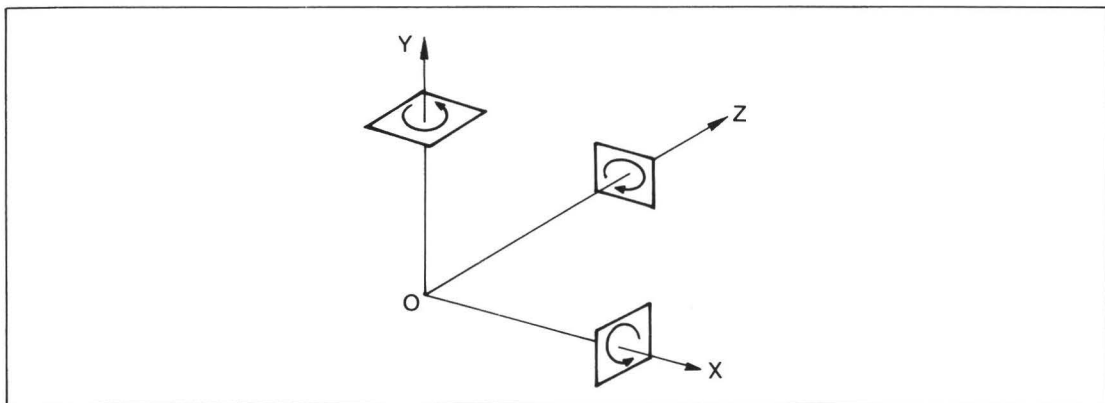



図1—15　左手系の回転

図形表示技術の基礎

コンピュータ内部に形状を記述する方法として，データ構造の種類について述べ，また，記述したデータの座標変換について解説しました。これらのデータを，コンピュータ上に図形として表示するためには，いろいろな考え方があります。

以下に示す3つの座標系と2つの変換で考えるのが便利で，一般的だと思います（座標系の名称は，種々の本によって統一されていません。内容は同一であるのに違っている場合が多いようです。本書では，以下のように統一します）。

(A)　マスター座標系

種々のアプリケーション・プログラムのインタフェースが行われる座標空間であり，この座標系上でセグメント（図形要素）を形成します。

(B)　ワールド座標系

個々の入出力装置（グラフィック・ディスプレイ装置等）とは独立した座標系であり，この座標系にそれぞれの物体を定義させておきます。マスター空間で定義したおのおののセグメントの相対的，絶対的位置関係を定義する空間です。

(C)　スクリーン座標系

入出力装置の物理的な座標系を指します。これによって，画面上で実際に見える空間が定義されます。

例えば，直方体A，Bは，形状も大きさも同一なので，マスター空間での定義は1つにします。ワールド座標系は，マスター座標系に記述された物体を，どこの位置に表示させるか規定したものです。

マスター座標系からワールド座標系への変換を，インスタンス変換といいます。この変換により，マスター空間で定義した対象物を，ワールド座標系の中の任意な位置に配置させます。ワールド座標系で定義した空間は，ワークステーション変換により，入出力装置に合わせて，どの方向からどの方向へ向いて眺めているかを定義します。

マスター座標系で表されるマスター空間と，ワールド座標系で表されるワールド空間は，ともに3次元空間であり，この本では右手系で表示しています。ところが，出力装置であるスクリーン座標系で表されるスクリーン空間は，2次元空間です。このために，3次元から2次元に変換する必要が生じます。これが，3次元2次元変換というものです。

通常，マスター座標系に定義した物体は，ワールド座標系の中で，位置，大きさ，傾き等の情報を付加されて配置され（この変換は，インスタンス変換と呼ぶ），私たちの通常の世界と同様の情報を持つ結果となります。位置情報等を持ったワールド座標系上に定義された物体を，何らかの方法で見る必要があります。これが，ワークステーション変換を行ったスクリーン座標系の表示モデルです。

ワークステーション変換は，表示するためのワークステーションに合わせて，また，表示したい図形に合わせてさまざまの変更を加える必要があります。ウインドウ変換，ステレオ変換，透視変換等，

その画面に合わせて変更し表示します。

スクリーン座標系は，表示装置または出力装置に固有の座標系であり，特殊な場合を除いて２次元による座標系であるといえます。そのため，座標系相互間で，原点位置，座標軸の方向，原点寸法等が異なることがあり，変換が必要とされます。

次項以下，座標系相互の変換技術について解説します。

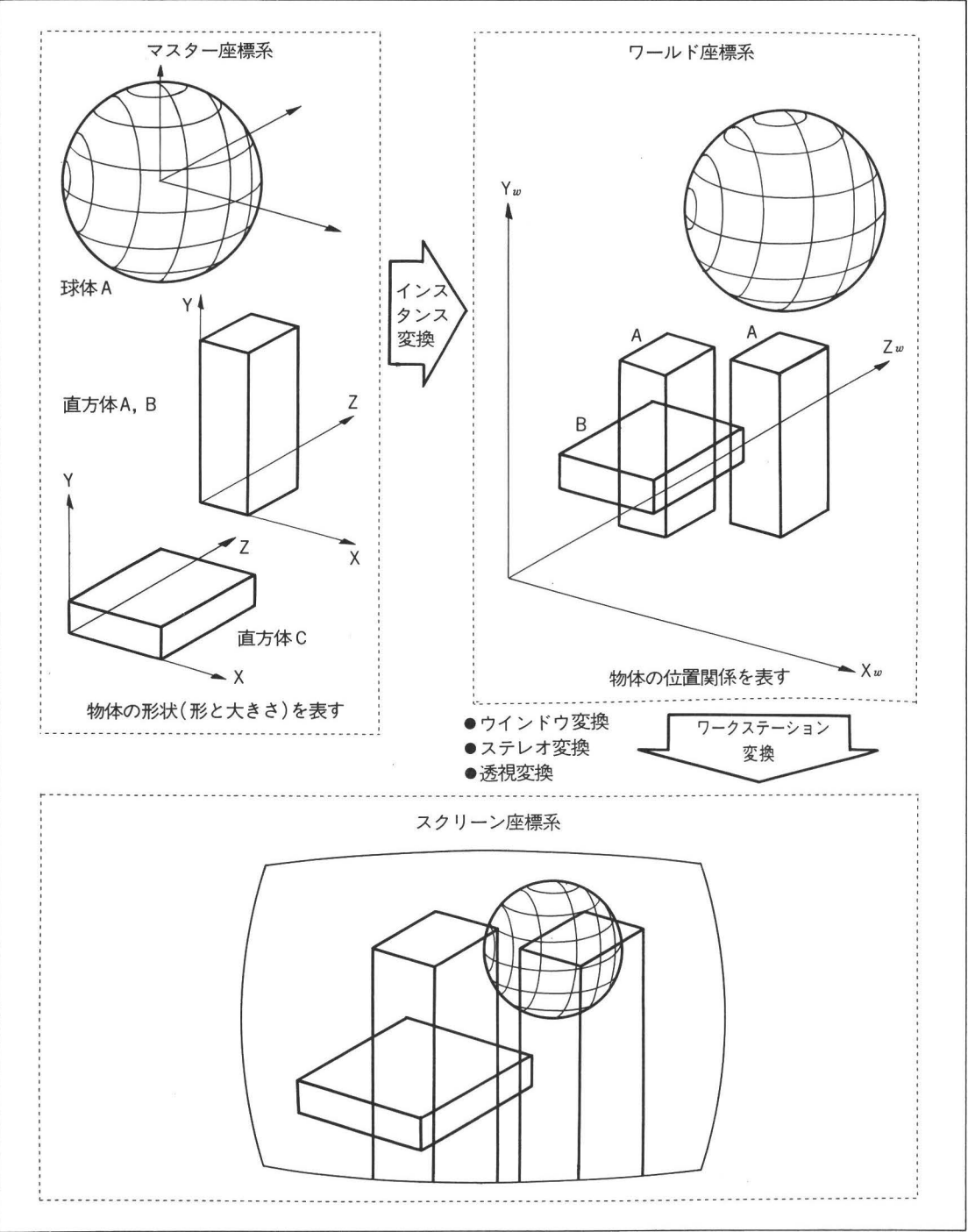



図１—16　３つの座標系と２つの変換

ウインドウ変換

座標系間の2次元的な変換をウインドウ変換（Window translation）と呼びます。一方の座標系にある長方形の領域を，他方の（変換後の）座標系の長方形の領域に移動する技術です。X1ターボ／X1シリーズの場合，この変換に関しては WINDOW 命令文が存在し，あらためて数値計算を組むまでもなくできるようになっています。

しかし，ウインドウ変換は，これらの便利さ以外に大がかりなシステムを組む場合には，1画面に複数のビューポートを設置する必要があるという，大切な要素を秘めています。ここでは，そういったことを踏まえて，ウインドウ変換の仕組みについて解説します。

ワールド座標系をスクリーン座標系に変換する場合について考えてみると，まず，ワールド座標系の中で表示したい座標領域を，左上端の点（W_e, W_t）と，右下端の点（W_r, W_b）で囲まれた領域とし，また，スクリーン座標系の表示座標領域（ビューポート）を左上端の点（V_e, V_t），右下端の点（V_r, V_b）で囲まれた領域と設定します。（W_e, W_t），（W_r, W_b），ワールド座標系の座標値，（V_e, V_t），（V_r, V_b）はスクリーン座標系の座標値であるといえます。

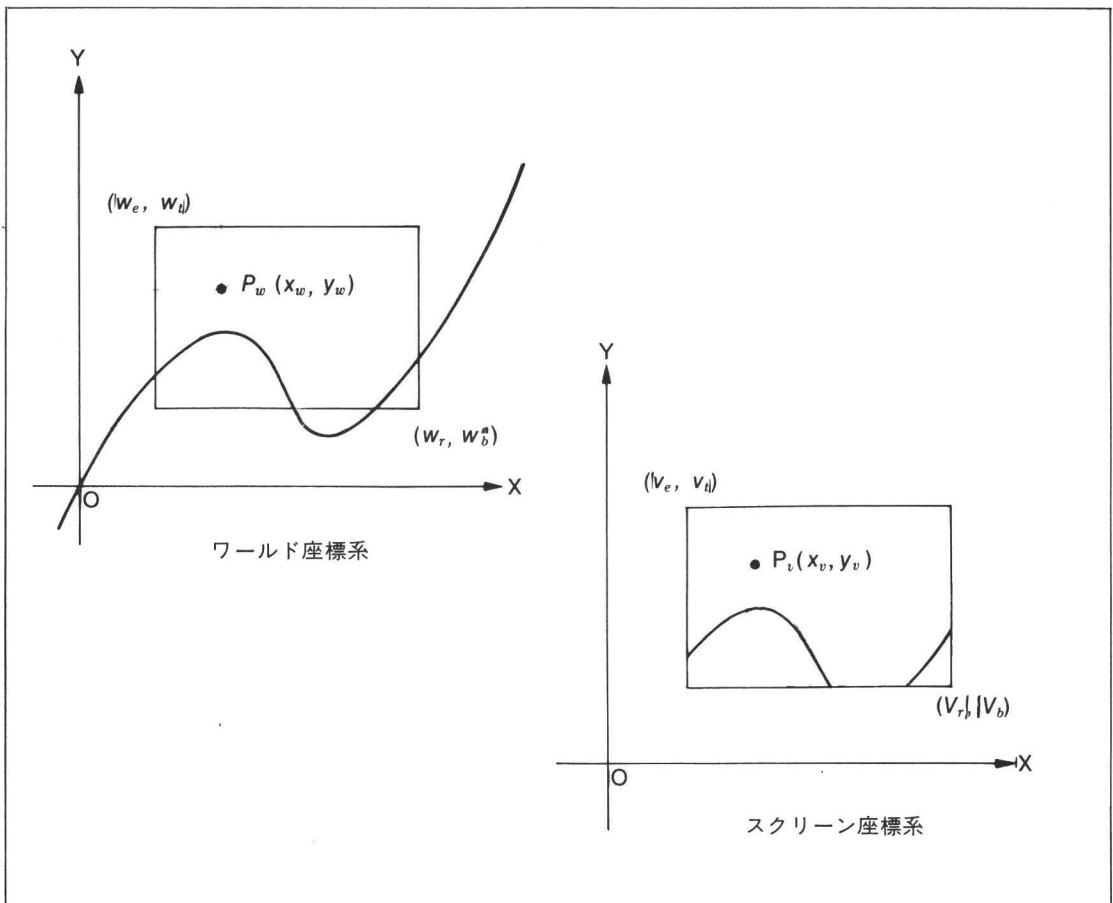



図1—17　ウインドウ変換

ワールド座標系のウインドウ内の任意の1点 P_w の座標を（X_w, Y_w），スクリーン座標系のそれに対応する点 P_v の座標を（X_v, Y_v）とすると，ワールド座標系のウインドウの横幅は，

$W_r - W_e$

点 P_w までの左端からの距離は，

$X_w - W_e$

と表せ，その比率は，次の式で表せます。

$$\frac{X_w - W_e}{W_r - W_e} \quad \text{——①}$$

同様にして，スクリーン座標系においてビューポートの大きさは，X方向の横幅を，

$V_r - V_e$

点 P_v までの左端からの距離は，

$X_r - V_e$

と表せ，その比率は，次のように書き表せます。

$$\frac{X_r - V_e}{V_r - V_e} \quad \text{——②}$$

①と②の2つの式には，次の関係式が成り立ちます。

$$\frac{X_w - W_e}{W_r - W_e} = \frac{X_r - V_e}{V_r - V_e} \quad \text{——③}$$

同様にしてY軸方向でも，次の関係式が導き出されます。

$$\frac{Y_w - W_b}{W_t - W_b} = \frac{Y_v - V_b}{V_t - V_b} \quad \text{——④}$$

③式，④式より，ビューポート上の点 P_v のX座標 X_v，Y座標 Y_v は，次の式で得られます。

$$\left.\begin{aligned} X_v &= \frac{V_r - V_t}{W_r - W_t}(X_w - W_b) + V_e \\ Y_v &= \frac{V_t - V_b}{W_t - W_b}(Y_w - W_b) + V_b \end{aligned}\right\} \quad \text{——⑤}$$

⑤式をあらためて書き直すと，次のように表現できます。

$$\left.\begin{aligned} X_v &= aX_w + b \\ Y_v &= cY_w + d \end{aligned}\right\} \quad \text{——⑥}$$

ただし　$a = \frac{V_r - V_t}{W_r - W_t}$，$b = V_t - \frac{V_r - V_e}{W_r - W_e} W_e$，$c = \frac{V_t - V_b}{W_t - W_b}$，$d = V_b - \frac{V_t - V_b}{W_t - W_b} W_b$

これをあらためてマトリックスで書き換えると，次のように書き表せます。

$$[X_v \quad Y_v \quad 1] = [X_w \quad Y_w \quad 1]\begin{bmatrix} a & 0 & 0 \\ 0 & c & 0 \\ b & d & 1 \end{bmatrix}$$

ただし　a，c：ウインドウからビューポートへの拡大率（または縮小率）

b，d：移動量

視点変換

3次元ワールド座標系内にあるモデルの見え方は，観察者の見る視点と注目点との位置によって異なります。先に座標変換の過程で，図形の定義はマスター座標系に定義しておき，あらためてインスタンス変換の後，ワールド座標系に定義すると述べました。

ここでワールド座標系に定義されているモデルは右手系の座標記述ですが，スクリーン座標系でこれを見るためには，コンピュータ・ディスプレイの2次元のスクリーン図形として変換する必要があります。このため，3次元ワールド空間の適切な位置に投影用の平面（すなわち投影面：plane of projection または view plane）を設ける必要が生じてきます。

ワールド座標系（$O_w-X_wY_wZ_w$）において，視点 P_f（X_f, Y_f, Z_f）と注目点 P_a（X_a, Y_a, Z_a）とを決めると，直線 P_fP_a をZ軸とした左手系の座標系（$O_e-X_eY_eZ_e$）が描けます。これを視点座標系といい，ワールド座標系で定義されたモデルの位置情報を，この視点座標系での位置情報に変換することを視点変換といいます。なお，この視点変換は，次項の透視変換のための前準備です。

ワールド座標系の中の点（X_w, Y_w, Z_w）を，視点座標系の点（X_e, Y_e, Z_e）に変換するマトリックス T_v を求めてみましょう。次の式で表すことができます。

$$[X_e \quad Y_e \quad Z_e \quad 1]=[X_w \quad Y_w \quad Z_w \quad 1]T_v$$

図1—18のように，X_e 軸を X_wZ_w 平面に対して平行にとると，X_e, Y_e, Z_e 軸の方向が自動的に決まります。ここでまず，原点 O_w を視点 P_f へ移動する変換マトリックスを考えると，次のように表

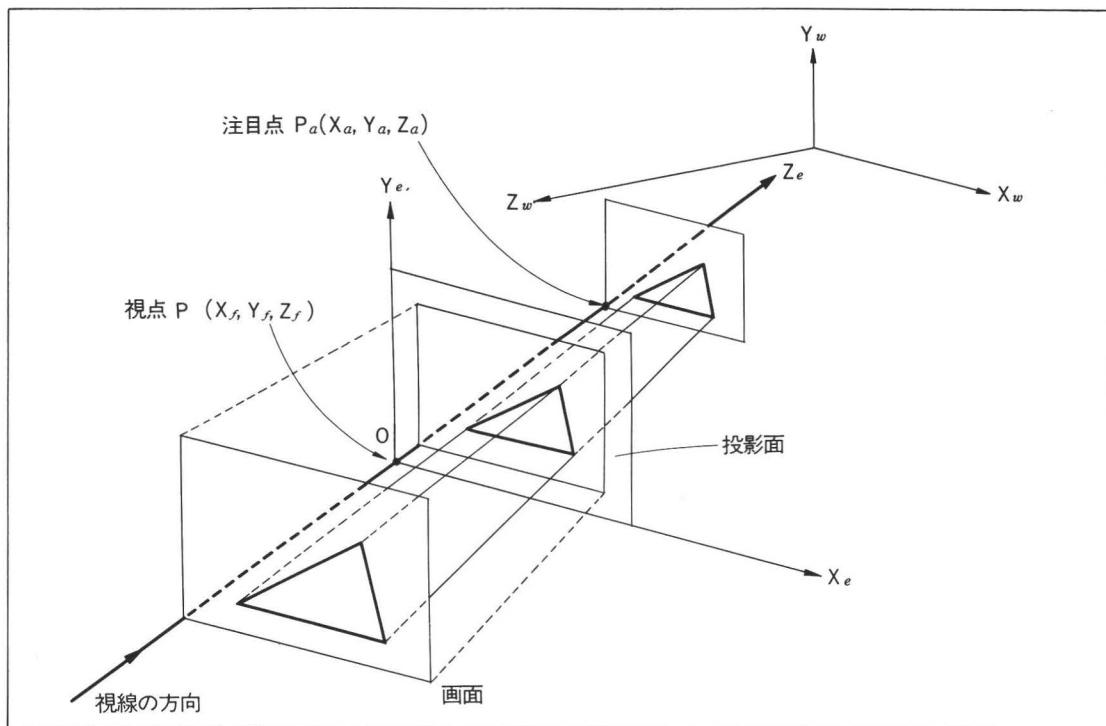



図1—18　3次元ワールド座標系と視点座標系

すことができます。

$$[x_1 \quad y_1 \quad z_1 \quad 1]=[x_w \quad y_w \quad z_w \quad 1]T_1$$

ただし　$T_1=\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -x_f & -y_f & -z_f & 1 \end{bmatrix}$

これにより，X_e および X_1 軸は，$X_w Z_w$ 平面に平行な平面上で，X_l と X_e 軸の原点は点 P_f となります。X_1 と X_e のなす角を α とし，座標系（$O_1-X_1Y_1Z_1$）を Y_1 軸のまわりに $-\alpha$ だけ回転して新しい座標系（$O_2-X_2Y_2Z_2$）を求めると，X_2, Z_2 軸と X_e, Y_e 軸とはそれぞれ一致します。

これを変換マトリックスで書き表すと，次のようになります。

$$[x_2 \quad y_2 \quad z_2 \quad 1]=[x_1 \quad y_1 \quad z_1 \quad 1]T_2$$

ただし　$T_2=\begin{bmatrix} \cos\alpha & 0 & -\sin\alpha & 0 \\ 0 & 1 & 0 & 0 \\ \sin\alpha & 1 & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

$$\cos\alpha=\frac{-(Z_a-Z_f)}{\sqrt{(x_f-x_a)^2+(z_f-z_a)^2}} \qquad \sin\alpha=\frac{x_a-x_f}{\sqrt{(x_f-x_a)^2+(z_f-z_a)^2}}$$

ここで Y_2 軸は，Y_eZ_e 平面上に存在していますが，今のところ Y_e 軸と角度 β だけずれているのがわかります。そのため，座標系（$O_2-X_2Y_2Z_2$）をX軸のまわりに $-\beta$ だけ回転させ，座標系（$O_3-X_3Y_3Z_3$）とすると，X_3, Y_3, Z_3 軸と X_e, Y_e, Z_e 軸は完全に一致します。これを変換マトリックスで書き表すと，次のようになります。

$$[x_3 \quad y_3 \quad z_3 \quad 1]=[x_2 \quad y_2 \quad z_2 \quad 1]T_3$$

ただし　$T_3=\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\beta & \sin\beta & 0 \\ 0 & -\sin\beta & \cos\beta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

$$\cos\beta=\frac{\sqrt{(x_f-x_a)^2+(z_f-z_a)^2}}{\sqrt{(x_f-x_a)^2+(y_f-y_a)^2+(z_f-z_a)^2}}$$

$$\sin\beta=\frac{y_f-y_a}{\sqrt{(x_f-x_a)^2+(y_f-y_a)^2+(z_f-z_a)^2}}$$

これで座標系（$O_3-X_3Y_3Z_3$）と（$O_e-X_eY_eZ_e$）との違いは，Z軸の方向だけとなり，その結果，次の式が成り立ちます。

$$[x_e \quad y_e \quad z_e \quad 1]=[x_3 \quad y_3 \quad z_3 \quad 1]T_4$$

ただし　$T_4=\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

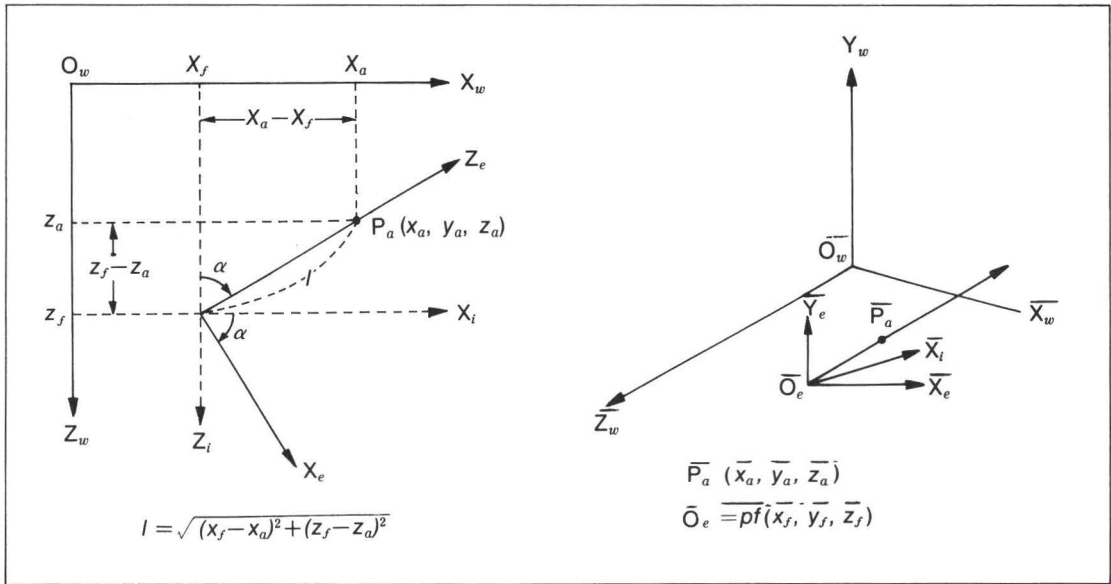



図1—19　座標軸の回転(XwZw面に垂直な方向の投影)

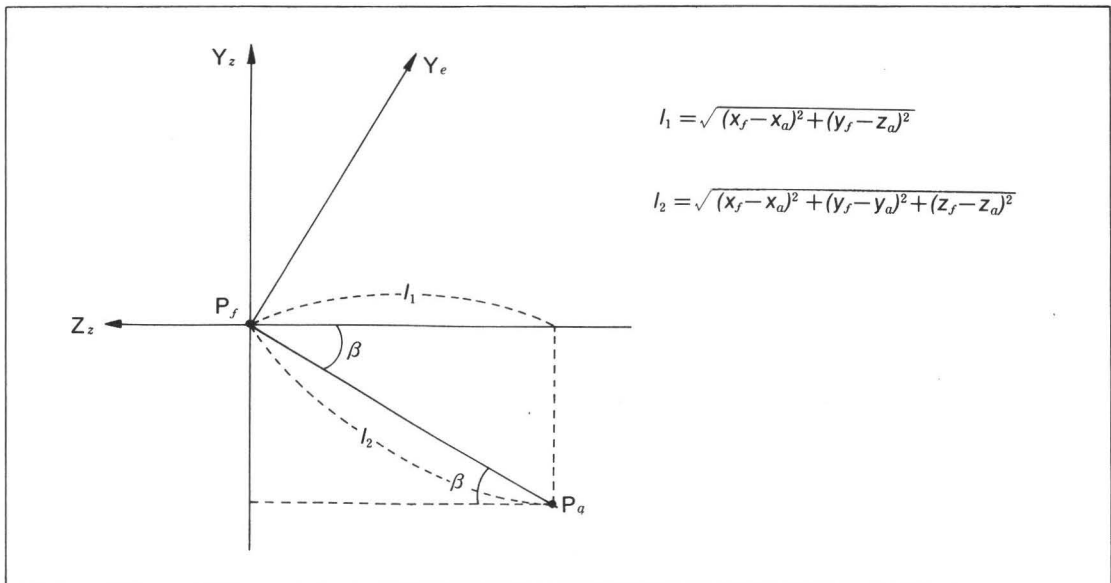



図1—20　座標軸の回転(YeZe面に垂直な方向の投影)

以上をまとめると，次のように表せます。

$$T_v = T_1 T_2 T_3 T_4$$

$$= \begin{bmatrix} \cos\alpha & \sin\alpha\sin\beta & \sin\alpha\cos\beta & 0 \\ 0 & \cos\beta & -\sin\beta & 0 \\ \sin\alpha & -\cos\alpha\sin\beta & -\cos\alpha\cos\beta & 0 \\ -x\cos\alpha - z_f\sin\alpha & \begin{array}{r} -x_f\sin\alpha\sin\beta - y_f\cos\beta \\ +z_f\cos\alpha\sin\beta \end{array} & \begin{array}{r} -x_f\sin\alpha + \cos\beta + y_f\sin\beta \\ +z_f\cos\alpha\cos\beta \end{array} & 1 \end{bmatrix}$$

ここで T_v は，ワールド座標系で記述されているデータを視点座標系に変換するマトリックスであるといえます。

透視変換

最後に，コンピュータ・ディスプレイ等の表示装置に表示するために，3次元空間に描かれている像を2次元図形に変換する必要があります。それらについて解説します。

最も単純なものは，観察者の視点の位置が無限遠点にあると仮定して，スクリーンに平行に投影する方法です。これは，平行透視と呼び，3次元図形の視点座標（X, Y, Z）のY成分だけを無視して，（X, Y）の座標値をスクリーン座標系上の点とすればよいのです。しかし，この方法は，設計製図のような寸法が重要となる場合においては有効だといえますが，たいていの場合は，3次元空間内のZ_{max}とZ_{min}の値の差が大きすぎるために，遠近感のない不自然な形態となってしまいます。

そのために，透視変換では，一般的に，私たちが実際のモデルを現実の世界で照らし合わせて見ることができるように，近くのモデルは大きく，遠方のものは小さくといった効果が出るように工夫されています。

ここでは，視点変換でも述べたように，視点を原点とし，各座標軸を左手系にとります。また，スクリーン座標を原点からhの距離に置き，3次元空間の1点P（X, Y, Z）と原点を結ぶ直線と，スクリーン平面との交点がスクリーン上の点P′の位置とすると，次の関係式が成り立ちます。

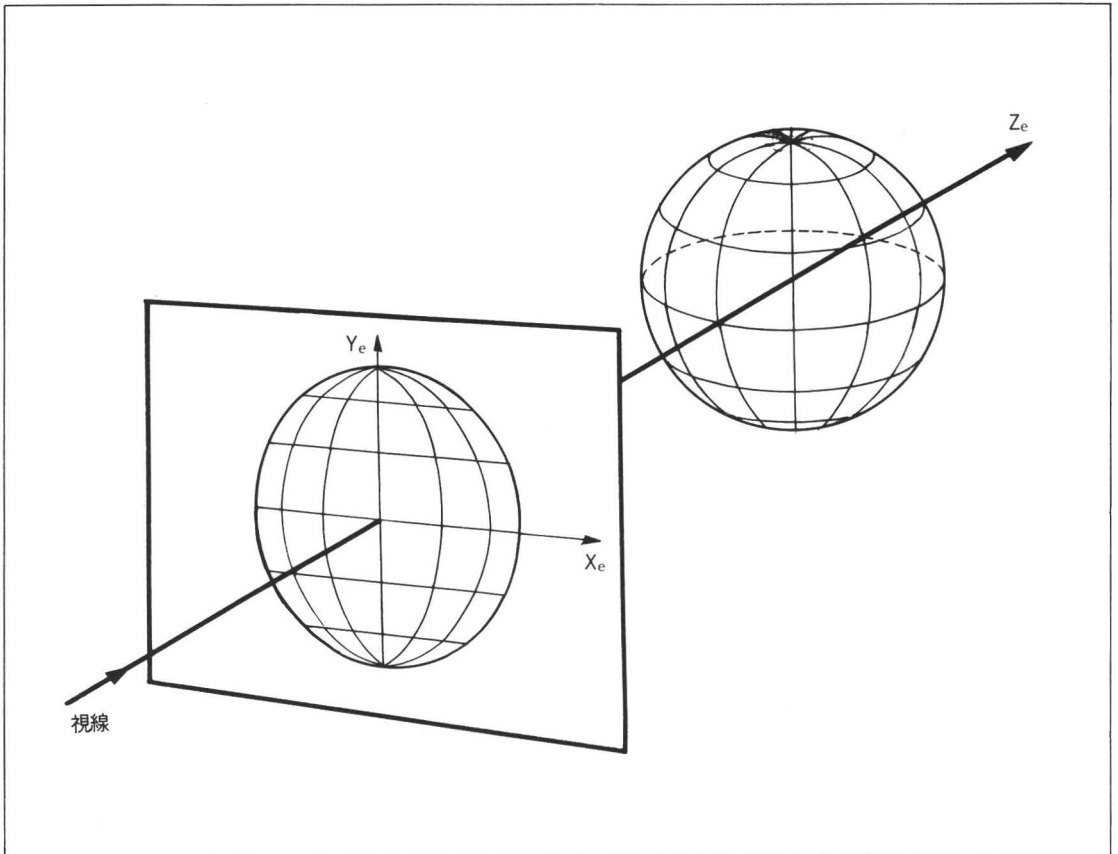



図1—21　透視変換

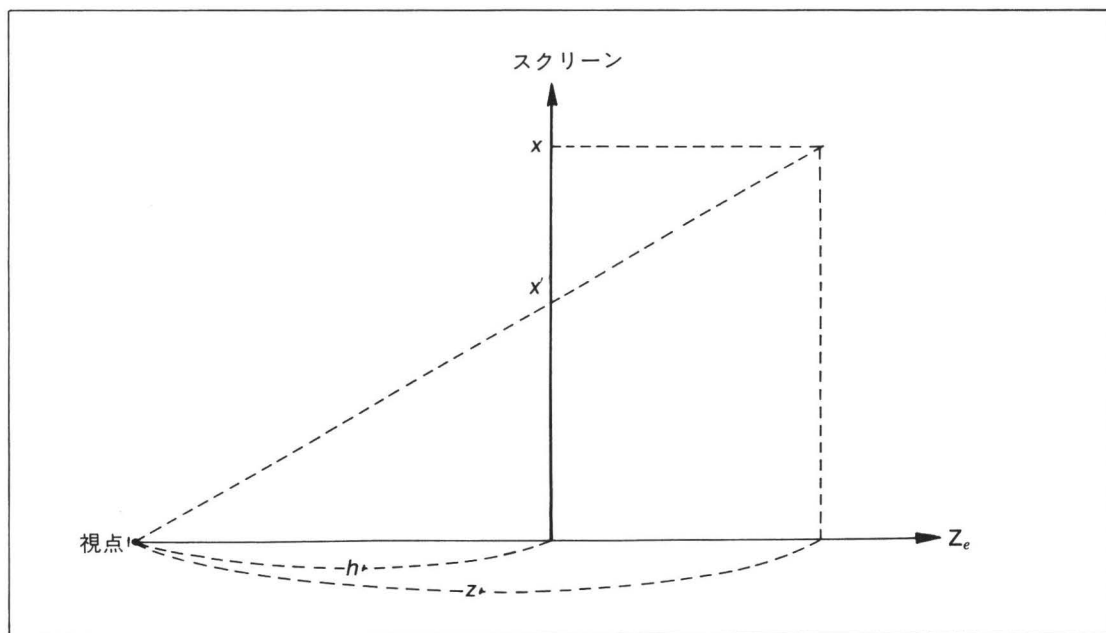



図1—22　透視変換における座標間の関係

$$x' = x \cdot \frac{h}{z}$$

$$y' = y \cdot \frac{h}{z}$$

これを変換マトリックスで書き表すと，以下のようになります。

$$[x' \quad y' \quad z' \quad w'] = [x \quad y \quad z \quad 1]\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 2 & \frac{1}{h} \\ 0 & 0 & -h & 0 \end{bmatrix}$$

ただし　$x' = x$　$y' = y$　$z' = 2z - h$　$w' = \frac{z}{h}$

ここで，$W' \neq 1$ であるから，左端のベクトルの各要素を W′ で割って，次のようになります。

$$x'' = x \cdot \frac{h}{z}$$

$$y'' = y \cdot \frac{h}{z}$$

$$z'' = 2h - \frac{h^2}{z}$$

これは，図1—22を参照していただくとわかるように，Z'' は奥行き方向の距離を示す値で，Zの値がhのとき $Z''=h$，Z の値が ∞（無限大）のとき $Z''=2h$ となり，$Z=h$〜∞ の距離を $Z''=h$—$2h$ へ圧縮した距離を表しています。

図形表現技術の基礎

前項までで，ワイヤーフレーム・モデルおよびサーフェイス・モデルでの図形記述について解説しました。これらはすべて，物体を表現するうえで大切な，形状を定義する仕組みの解説であったと思われます。この項では，一歩進んで，これらの物体を定義した後に，よりリアルに表現する手法について解説します。

この技術は，最近では，コンピュータ・グラフィックスによるシミュレーション技術として注目を集めています。本書では，これらの技術のうちマッピング，レイトレーシングについて基礎的なプログラミングを掲載してあります。その他の技術については，機会があれば別著にて紹介したいと考えています。ここでは，それらの表現技術についての概念を解説します。

一般に，コンピュータによる演算によって，グラフィックス・ディスプレイ上に視覚情報（シミュレーション・モデル）を生成する場合に最も基礎的なモデルは，シミュレートされるモデルの位置，光源の方向と強さ，観察者の位置，スクリーンの位置等が設定された後に，スクリーン上のピクセルからどのくらいの光量が，モデルから観察者の目に入るかを調べ，そのピクセルをそれぞれの明るさに決定するという形式をとります。これを実現するために，物体表現の特性に応じて，観察者の目に入る光量をスクリーン分布させる技術が必要となるが，表現技術としては次の４点が代表とされます。

シェイディング

シェイディング表現技術は，今日のグラフィックスによるシミュレーション表現の基礎になった理論だといえます。ディスプレイ上に表現するモデルは，サーフェイス（面）データを持たせ，サーフェイス・データを構成する多面体の特定の点への光の反射を計算させて表示します。これは反射のメカニズムのモデル化といえます。

シェイディングの方法は，ランバート・シェイディング（Lambert shading），グーロー・シェイディング（Gouraud shading），フォング・シェイディング（Phong shading）と，それぞれのアルゴリズムの開発者にちなんで，３種の方法があります。

ランバート・シェイディングは，ファセテッド・シェイディング（Faceted shading）とも呼ばれ，モデルを記述する多面体の中から単一の多面体を取り出し，その多面体に（残りの多面体に関係なく）陰影づけを行い，その多面体の計算が終了すれば次の多面体に移り，それを繰り返すという方法をとります。

この手法は，最も簡単でスピーディな方法ですが，表現モデルが写実的でなくごつごつしたものとなり，現在ではモデルの形や動きをすばやくチェックしたい場合に利用しています。

この表面のごつごつした感じをなくし，より滑らかな表現を可能にしたのが，グーロー・シェイディングです。別名スムーズ・シェイディング（Smooth shading）とも呼ばれ，多面体の中心によって反射光を計算させるのでなく，個々の頂点で計算させるという形式をとっています。

これは各面の陰影の輝度の平均をとって，対象物が滑らかであるという錯覚を起こさせます。その

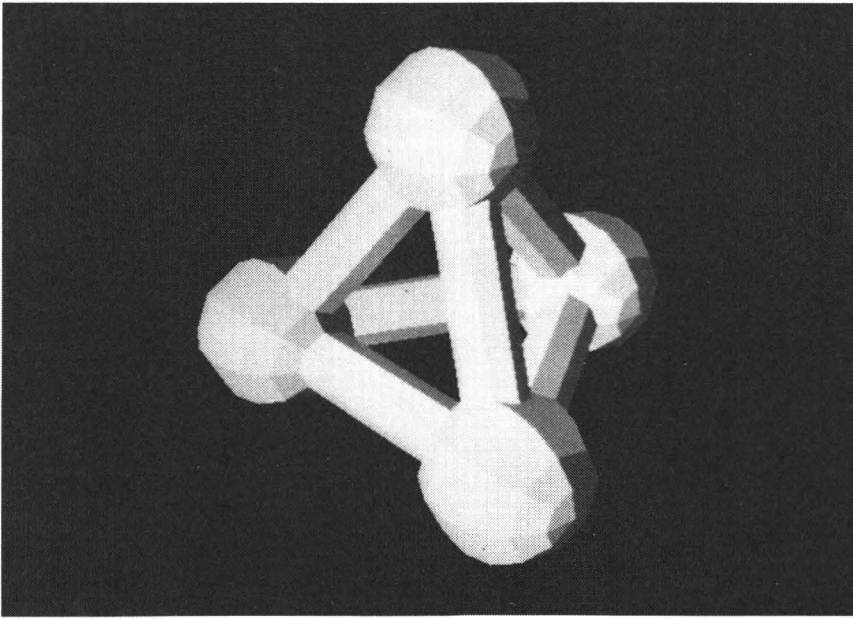



図1—23
ランバート・シェイディングによるモデル

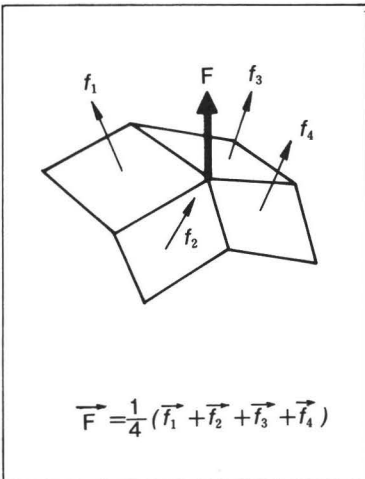



図1—24　グーロー・シェイディング

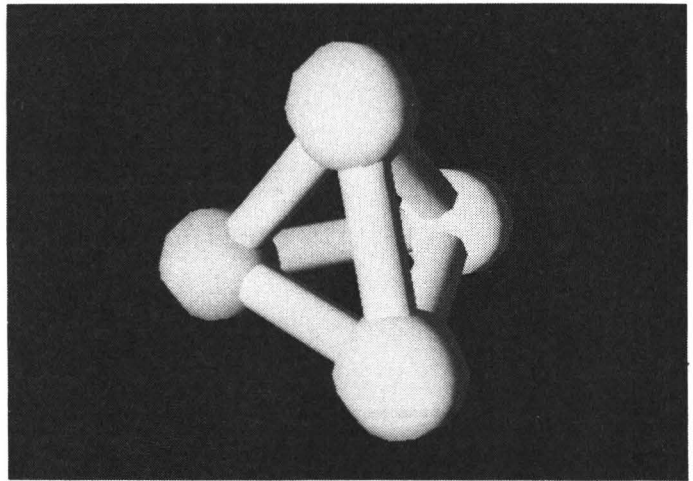



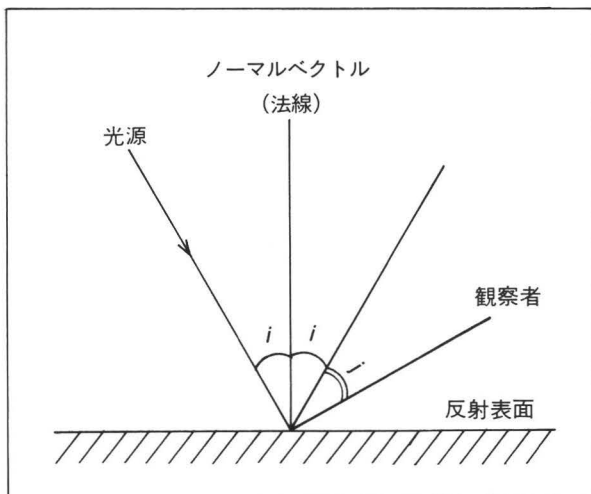
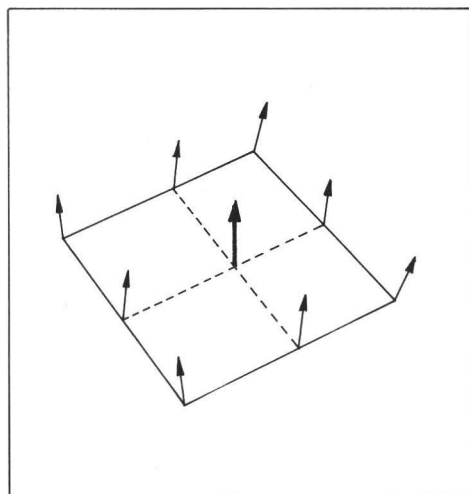
図1—25　グーロー・シェイディングによるモデル

ために，ハイライト面（光が強く当たっている面）等の表現の際に，頂点にたまたまハイライトの光源が当たればよいのですが，頂点からずれた場合にはうまく輝やかないということが生じる欠点があります。

フォング・シェイディングでは，グーローの方法での欠点を補うために，おのおのの頂点での法線ベクトルを求め，その法線ベクトルを多面体上での点に対して補完するという方式をとります。

この方式では，多面体形上のすべての点における法線ベクトルを，図1—24にあるように，まず，頂点での法線ベクトルを個別に求め，次に辺の中点で頂角での法線ベクトルの平均値をとり，最後に中心点で中点での法線ベクトルの平均値をとり，それに充てるという形式をとります。この法線ベクトルは，光源ベクトルや観察者の視線ベクトルと比較して表すということになります。

フォングのモデルは，次のように表せます。

$$I=I_a+\sum_{l} I_r l+I_t$$

図Ⅰ—26　フォングのモデル

図Ⅰ—27　フォング・シェイディング

I：観察者の目に入る光の強さ＝Pixel の明るさ
I_a：アイマイ光源による反射光の強さ
$\sum_l I_r l$：光 l による反射光の強さの和
I_t：物体を通過してくる光の強さ

$$I_r l = \{kd \cos i + ks(\cos j)^n\} \cdot I_l$$

I_l：光源 I の強さ
i：光源 I の入射角
j：光源 I の反射軸と観察者のなす角

kd, ks, n：物体表面の鏡面度を表す係数

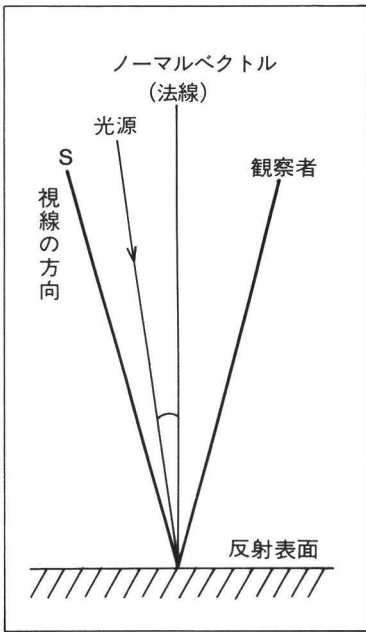
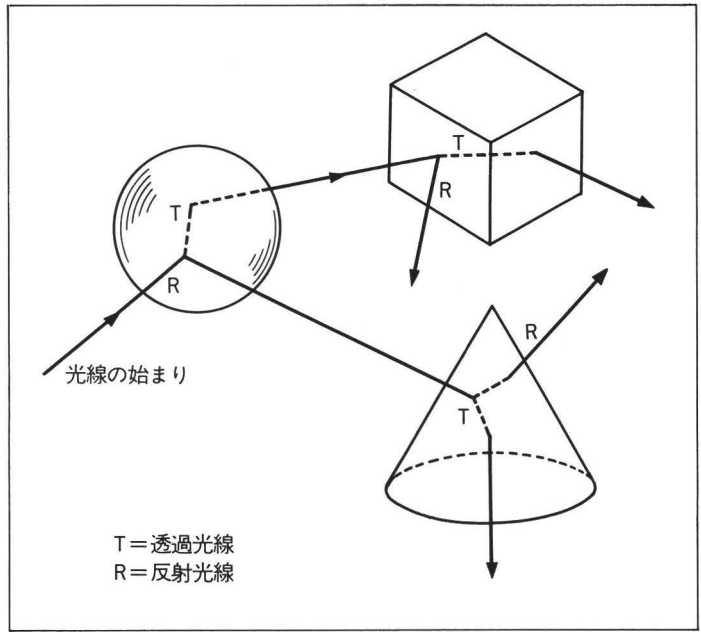
この手法は，計算に多くの時間を必要としますが，現在最もリアルな手法として使用されています。しかし，正反射光を三角関数のべき乗で近似した場合に，プラスチック風の材質感になってしまうというきらいがあります。

レイトレーシング

レイトレーシング（Ray tracing）表現技術は，シェイディングの手法が，物体表面上の点に入射する光がどのくらい観察者の方向に向かうのかを計算させたのに対し，まったく逆に，観察者の目から始まり画面に入る直線を追跡し，それが当たる対象物を調べるという方式をとるものです。この方式は，光学の法則に忠実に基づいているため，最も写実的な描画法といえますが，逆に最も時間を要する手法でもあります。

まず，観察者の目から出た直線を追跡し，対象物に当たった場合には，その光線を表面ではね返らせ，最初の対象物に何が反射されているかを調べます。対象物が透明度を持ったものであれば，対象物表面ではね返らさせるだけでなく，もう１本内部に光線（通過光線）を送り追跡していきます。

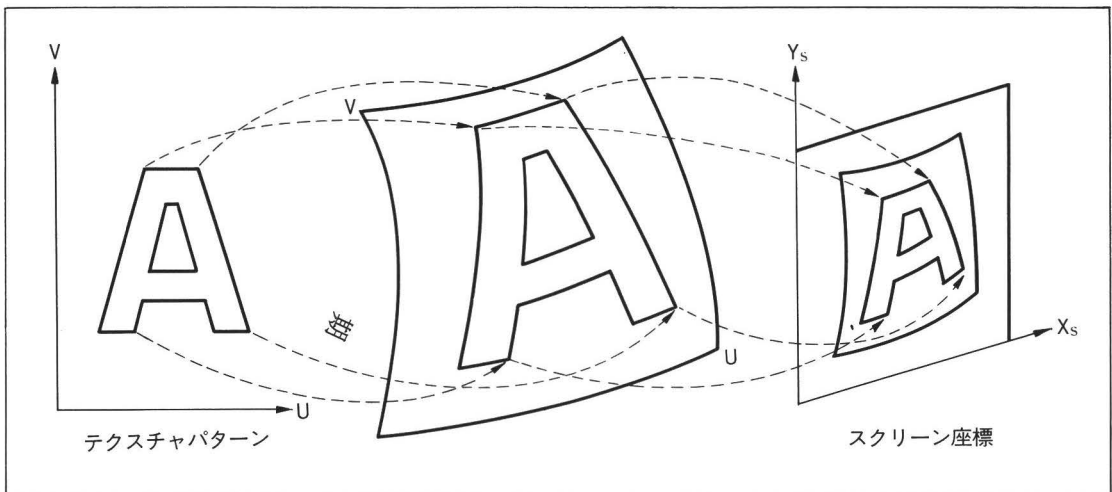
通過光線は，対象物に応じた境界の屈折率に影響されます。さらに追跡していき別の対象物に当たった場合には，同じプロセスが繰り返されることとなります。このように光線をたどっていくと，それらの陰影がわかるわけです。

この手法については，第５章で解説していますので，そちらを参照してください。

図Ⅰ—28　レイトレーシングのモデル

図Ⅰ—29　レイトレーシングのしくみ

テクスチャ・マッピング

実モデルの世界では，ほとんどの対象物に模様や材質感等のテクスチャが付いています。このテクスチャを表現するために，２次元のデータを作成し３次元モデルにはりつけるという手法があります。これをテクスチャ・マッピング（Texture mapping）と呼んでいます。

２次元の図形は，２次元による入力のほか，スキャナ，テレビカメラ等で読み込み，３次元モデルの表面にマッピングしてディスプレイ上に表現します。

図１—30のように，U—V座標系に取り込まれたテクスチャ・パターンを，パラメトリック曲面上のU—V座標系に写像し，それに前記のシェイディングを施し，スクリーン画面上に表示すれば画面上に処理された画像ができます。この手法については第４章で詳細に解説します。

図Ⅰ—30　２次元マッピング

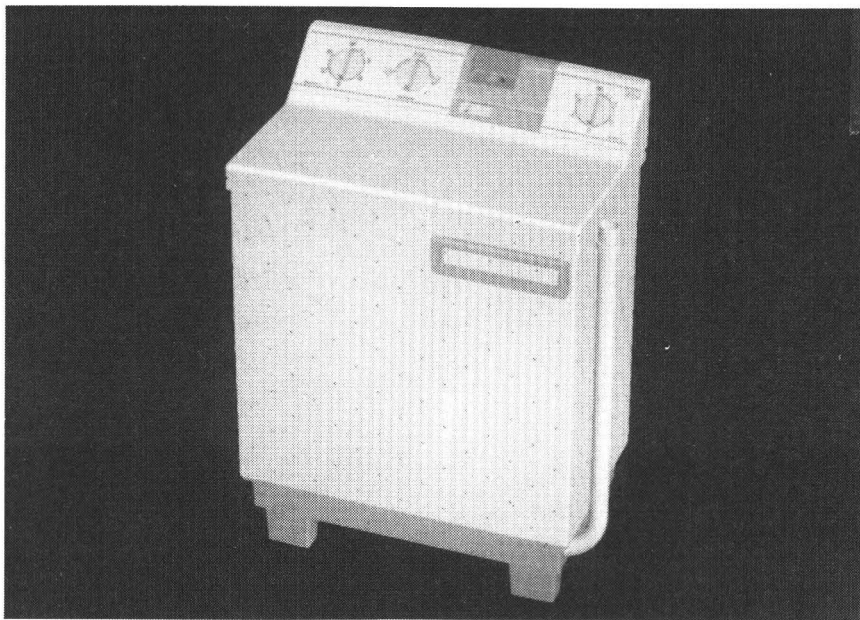



図1—31
2次元マッピングされた
洗たく機の操作パネル

バンプ・マッピング

2次元のパターンではなく，表面が細かい凹凸のテクスチャーには，バンプ・マッピング（Bump mapping）の手法が有効です。

基本的には，表面に凹凸がある場合には，サーフェイスを相当な数のパッチに分割して形状入力するのが正式な方法であるが，入力の手間，多大な計算時間を含めて大変な労力を要します。そのため，2次元のテクスチャ・パターンをはりつけ表現する前に，面のテクスチャを使ってそれが適用される面を修正し，これによって面に凹凸感を表現するものです。

これは，対象物上のピクセルを選び，その位置の法線ベクトルを求め，次にバンプ・マッピングを行う点を探し，その法線ベクトルを得て，最後にこの法線ベクトルを摂動することによって，各ピクセルに陰影をつけるというプロセスを踏みます。図1—33は，電気スタンドのアーム部分にバンプ・マッピングを応用させています。

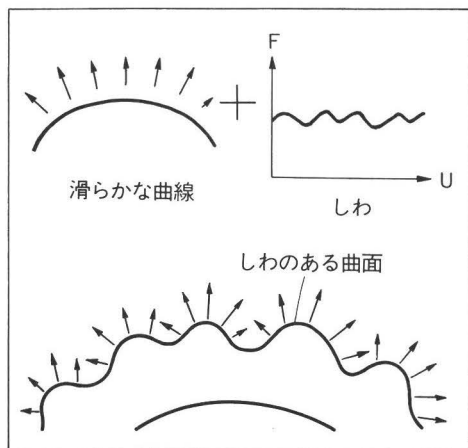



図1—32　法線ベクトルの摂動によるマッピング

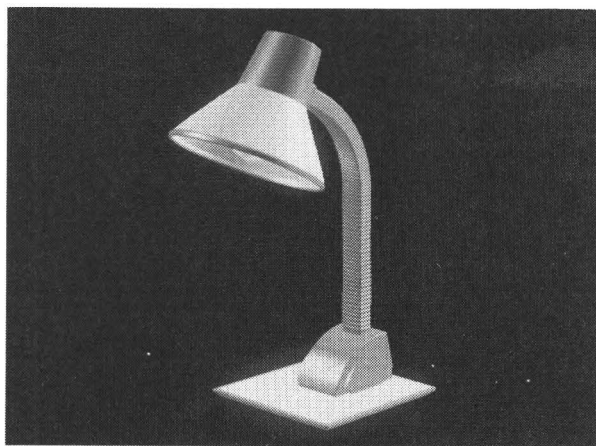



図1—33　電気スタンドのアーム部分へのバンプ・マッピング

変換マトリックスのまとめ

最後に，本章で解説してきた変換マトリックスについてまとめておきます。

3次元の変換を，変換マトリックスTで示すと，一般的に，

$$[X' \ Y' \ Z' \ 1]=[X \ Y \ Z \ 1]T \qquad (31)$$

と書き表すことができます。

ここで3次元変換マトリックスは4×4の要素から成り立っており，

$$T=\left[\begin{array}{ccc:c} A & D & G & M \\ B & E & H & N \\ C & F & I & O \\ \hdashline J & K & L & P \end{array}\right] \qquad (32)$$

ただし　J，K，L：x軸，y軸，z軸に関する移動（movement）を制御する要素

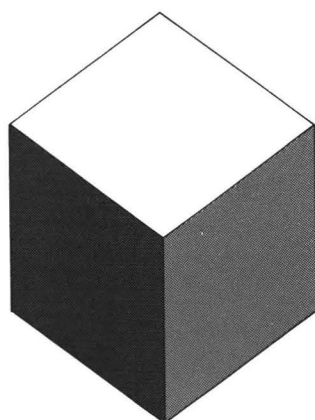
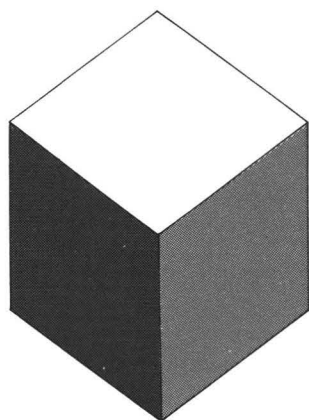
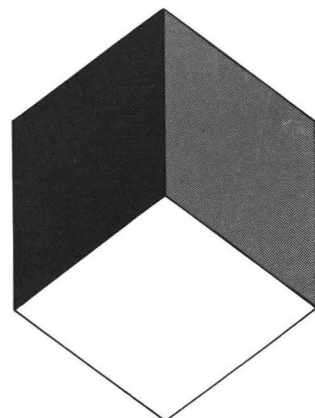
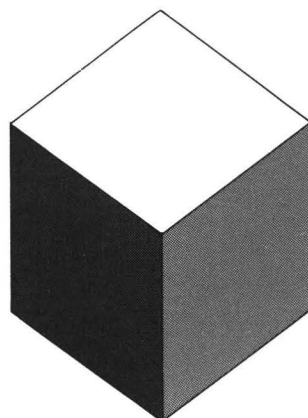
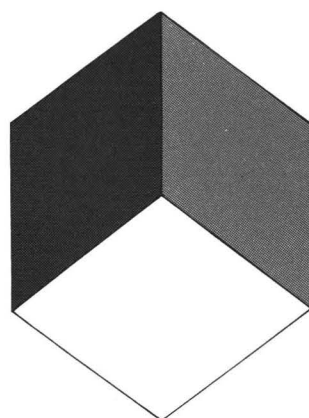
A，E，I：x軸，y軸，z軸に関する拡大／縮小（scaling）を制御する要素

B，C，D，F，P，I：x軸，y軸，z軸に関する回転（rotation），せん断などを制御する要素

M，N，O：x軸方向，y軸方向，z軸方向に奥行をとった透視変換を制御する要素

P：図形全体の拡大／縮小（scaling）を制御する要素

と書き表せます。

紙面の都合で簡単に変換技術について解説しましたが，この変換理論については，種々の良書が出ていますので，それらを参考にしてください。本書の以下のプログラムについては，これらの理論に基づいて組んであります。フローチャート，プログラムリストを参照しながら，3次元グラフィックスを自分のものにしてください。

2次元画像処理技術入門

デザインツール・プログラム

このデザインツールは，基本コマンドを用いて画面上にグラフィックスを作り出すものですが，描いた図形はグラフィック画面そのものに記憶させることになります（コンピュータ・メモリー上にデータは持っていません）。

そのため，画面上のピクセルがどの色かが，画像データとなります。基本的には，スキャナーやカメラ等が接続され，入力される状態となった場合は，同様の処理となります。

この方法は，コンピュータ・メモリー上に画像データを記憶する必要がないため，メモリー全部をコマンドに使用することができ，パソコンのような小規模なメモリー容量の機種にはうってつけの方法といえます。ただ，画面上のカラー情報を画像メモリーとして使用している関係上，拡大・縮小がドットごとの大きさとなったり，画像変換に関しては，そのたびごとに画素のカラー情報を読み出して処理するために，時間がかかるという欠点があります。

これは，画面上のピクセル情報を利用して画像を作成しており，後述のように，メモリー上にXY座標値を持っていないことから画像処理といい，それにより作られた画像を変換させる技術を画像処理技術といいます。

ここでは，カーソル移動による画像作成ツールを紹介します。

デザインツール・プログラムの操作方法

このプログラムを入力して実行（RUN）すると，画面上に，図2—2の表示が出て，このプログラムのデータ入力媒体を，(1)キーボードで行うのか，(2)ジョイスティック1で行うのか，(3)ジョイステ

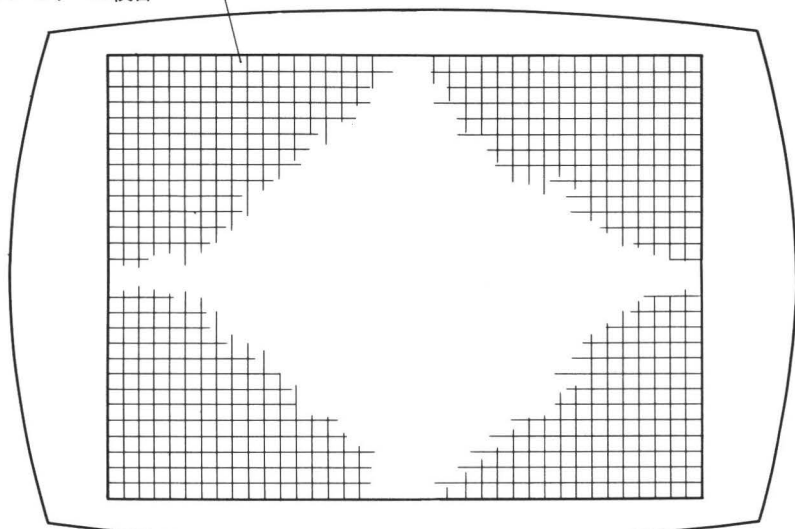



図2—1　画像処理

ィック2で行うのか聞いてきます。

(1)のキーボードで行う場合は，1を入力してテンキーに配置された指示に従って操作すると，画面上の十字カーソルが動くことになっています。

(2)および(3)のジョイスティックで行う場合は，CPUの裏面の2個のjoystick端子にそれぞれ別売のジョイスティックを購入し，差し込んでおく必要があります。

ジョイスティックの場合は，十字カーソルの動く方向にグリップを倒し，トリガーボタンを押すことにって PEN UP させたり，PEN DOWN させたりします。入力装置の選択を終えると，いよいよメインメニュー表示（図2—4）となります。

メインメニューは，1. INITIALIZE, 2. DISPLAY MODE, 3. SAVE, 4. LOAD, 5. NORMAL END の5つから成り，番号を入力することによってそれぞれのメニューを呼び出しま

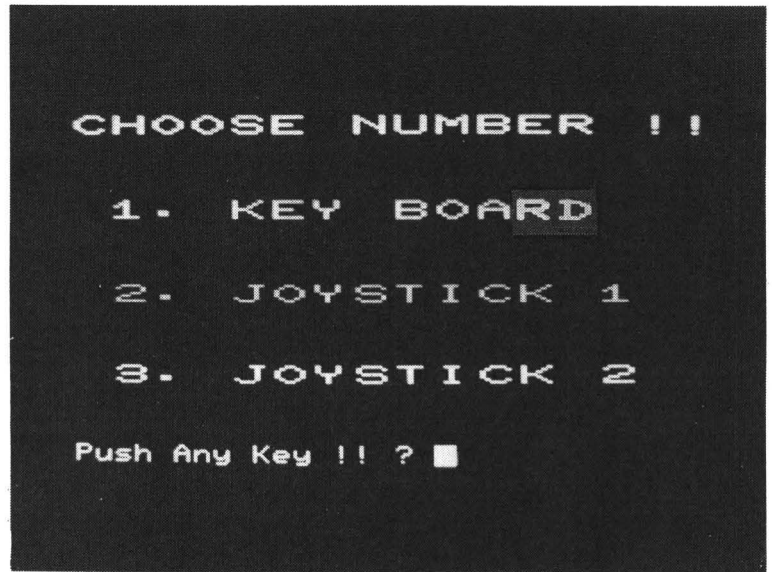



図2—2　入力装置の選択メニュー

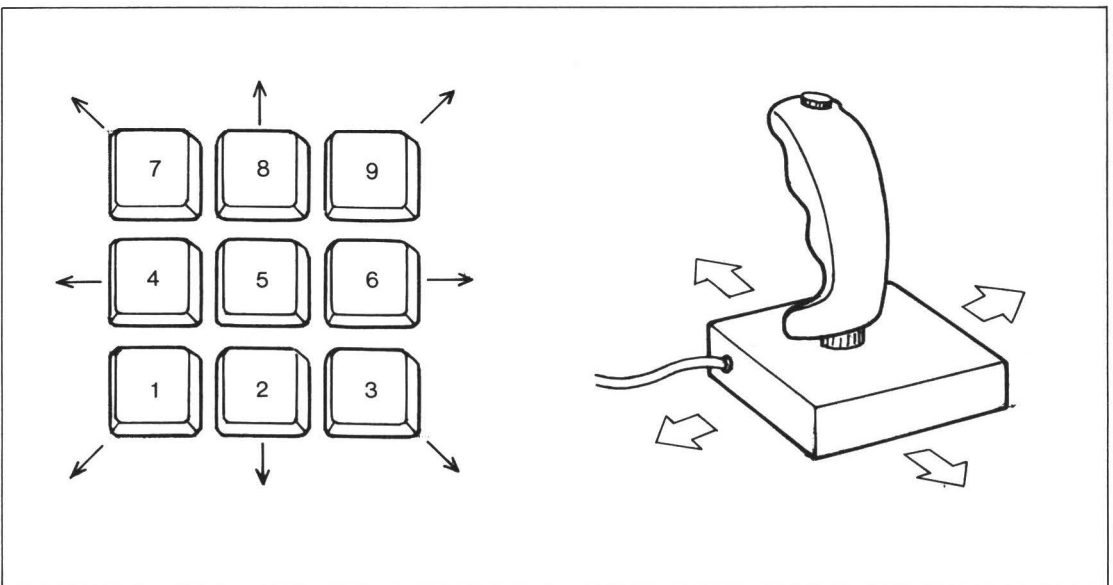



図2—3　キーボードとジョイスティック

す。５を選ぶと終了します。

```
[ DISPLAY TOOL MENU 2 ]

 1. INITIALIZE
 2. DISPLAY MODE
 3. SAVE
 4. LOAD
 5. NORMAL END

Select Number =
```

図２－４　メイン・メニュー表示

INITIALIZE

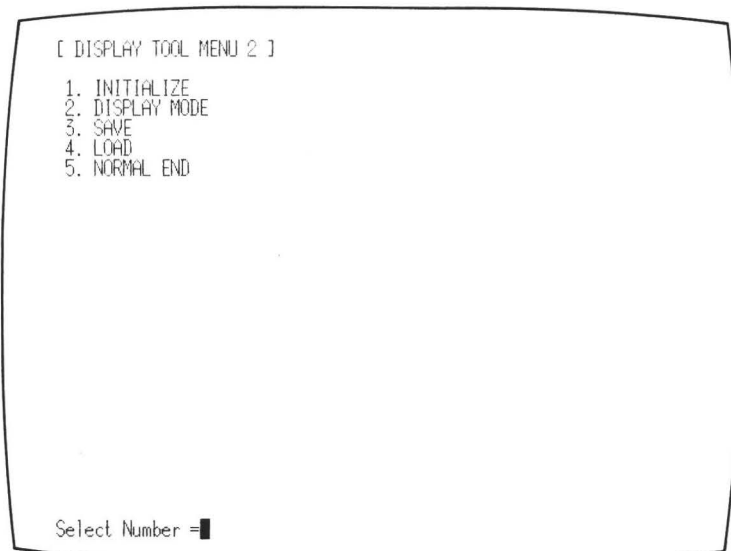



図２―４の状態で［1］キーを入力すると，図２―５が画面に表示されます。

１～４は画面密度の設定です。WIDTH 40の場合は，２つ画面があるので，必要に応じて 4. SCREEN CHANGE を使用してください。5. SCREEN CLEAR は，画面上の絵を消去し，6. EXIT は，図２―４のメインメニューに戻します。ここでも番号の入力で，それぞれ作動します（ただし，後述のマッピング処理を行うための画像入力は，［2］キー入力にて SAVE した画面のみで使用できるのでご注意ください）。

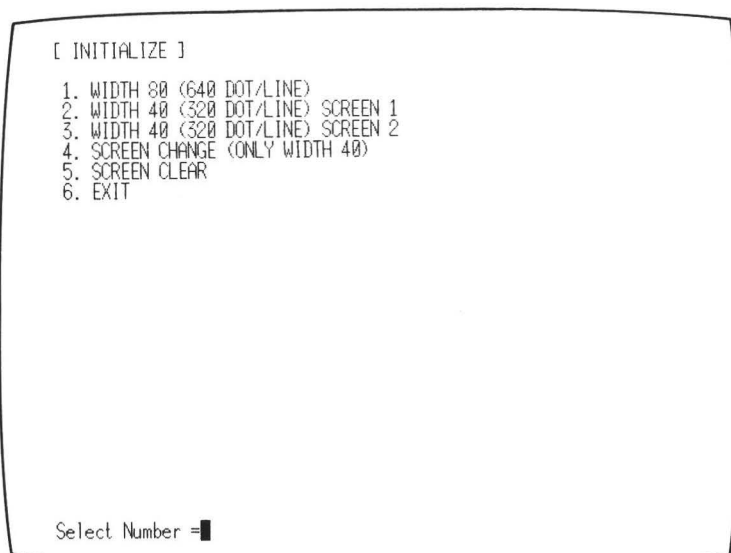



図２－５
INITIALIZEメニュー表示

DISPLAY MODE

メインメニューから，2. DISPLAY MODE を呼び出すと，図２―６が表示されます。画面下のアルファベットをコマンドと見なして入力すると，それぞれの命令が実行されます。

以下，それぞれの命令について概説します。

●B: BACK

B キーを入力すると，バックカラーを指定できます。下部に，カラー表示（0：黒，1：青，2：赤，3：マゼンタ(紫)，4：緑，5：シアン(水色)，6：黄，7：白）が出るので，この表示を確認しながら背景色を指定してください。

●Pp: PAINT

P キーを入力すると，カーソルを囲む閉曲線内を，Cコマンドで指定した色（カーソルの色）で塗りつぶします。

境界線の色は黒を除く7色です。境界線が完全に閉じていないと色がはみ出してしまいますので注意してください。

●U: PEN UP

U キーを入力すると，カーソルを移動しても画面に軌跡は描かれません。カーソルをペンに見たてると，ペンを紙から離している状態になります。

●C: COLOR

C キーを入力すると，カーソルで描く線やPAINTの色を指定できます。下部にカラー表示（0：黒，1：青，2：赤，3：マゼンタ(紫)，4：緑，5：シアン(水色)，6：黄，7：白）が出るので，カーソルの色を指定します。

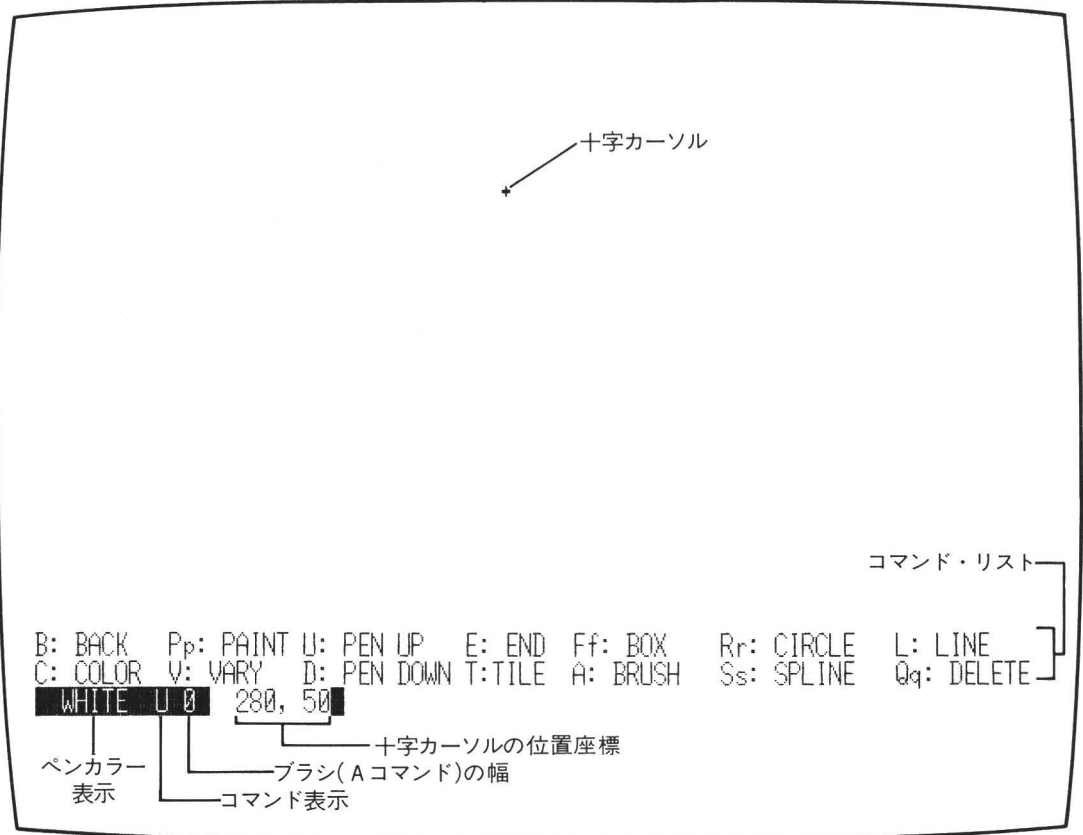



図2—6　DISPLAY MODEの表示画面

●V: VARY

[V]キーを入力すると，描いた絵のカラーパレット・チェンジができます。

●D: PEN DOWN

[D]キーを入力すると，カーソルの軌跡に線が描かれます。ペンを紙につけた状態です（ジョイスティックのトリガーボタンを押しても，同様の状態になります)。

●F: BOX

最初に[F]キーが押されたときのカーソル位置と，次に押されたときのカーソル位置を対角とする長方形を描きます。

●f: BOX（ペイント）

[F]キーを[f]キー（小文字）にすると，長方形内部をペンカラーの色で塗りつぶします。指定の方法はFと同様です。

●R: CIRCLE

最初に[R]キーが押されたカーソル位置を中心に設定し，次に[R]キーが押された位置から中心までの距離を半径とする円を描きます。さらに点を移動し[R]を押すと，その点を円周に含む円を描きます。

●r: CIRCLE（ペイント）

[r]キー（小文字）で入力すると，円の内部をペンカラーの色で塗りつぶして描きます。

●L: LINE

最初に[L]キーを押したときのカーソル位置を始点と設定し，カーソルを移動して次に[L]キーを押したときのカーソルの位置を終点として直線で結びます。引き続いてカーソルを移動して，[L]キーを押すと前の点から連続して直線を引くことができます。

●Ss: SPLINE

3点以上の点を指定し，その間を曲線で描くときに使用します。[S]キーを押し，閉曲線（ループ）ならば[1]キー，開曲線（ノンループ）ならば[0]キーを指定します。その後，カーソルを動かし，[S]キーを押すごとに点が指定されます。3点以上の点を指定後[s]キー（小文字）を押すと，[S]キーで指定した点の順に滑らかな曲線で結びます。閉曲線の終点→始点の間も結ばれます。

●Q: DELETE

操作ミス等によって，誤った位置に点を指定した後，これを取り消すのに使います。L, F, f, R, rの機能を解除するときに[Q]キーを使用します。なお以上のコマンドは，Qコマンドを使わなくても，U, D, X, Z, C, B, P, p, Tを除く他のコマンドを使用すると，自動的に解除されます。

●q: DELETE

L, F, f, R, rのコマンドのとき，[q]キーを入力すると，1つ前に実行した部分を解除します。

●T: TILE

P, pコマンドと同様，ペイント機能ですが，8色に限らず，色を複数の色の混合色で設定できるので，中間色やタイルパターンの表現に使用します。[T]キーを押すと，色の組み合わせパターンを聞いてくるので，数字で色を入力していき（例：2　4　5)，最後に[CR]キーを押します。すると，指定された組み合わせによるパターンが最下行に表示され，Yes, Noで確認を求めてくるので，[Y]

キー，Nキーのいずれかを入力します。Yキーを押すとペイントが実行され，Nキーでは色の組み合わせを再入力することができます。終了したい場合には，Eキーを押すと何も実行せずにメニューに戻ります。

●A：BRUSH

エアーブラシのような絵が描けます。Aキーを入力すると，ブラシの細かさ（0～8のドットの数）を聞いてきます。例えば，8キーを入力すると，Dと同じ使い方になります（Dの1ドットが8ドットで出る）。

●E：END

Eキーを押すとメインメニューに戻ります。

SAVE

メインメニューからSAVEを呼び出すと，図2―7が表示されます。ファイルネーム＋CRキーで，描いたグラフィックの画面をSAVEすることができます。

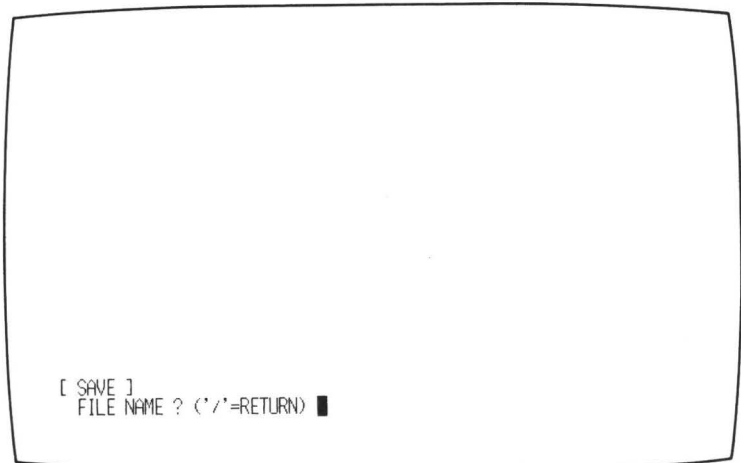



図2―7　SAVE時の画面

LOAD

メインメニューからLOADを呼び出すと，図2―8が表示されます。ファイルネーム＋CRキーで，SAVEしていたグラフィックの画面をLOADすることができます。

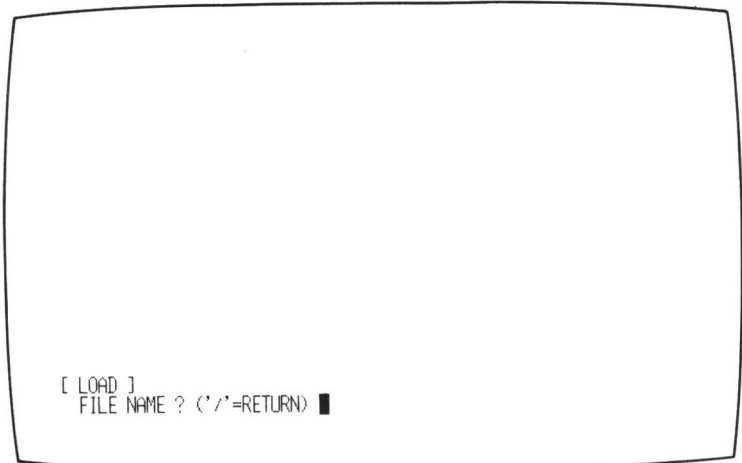
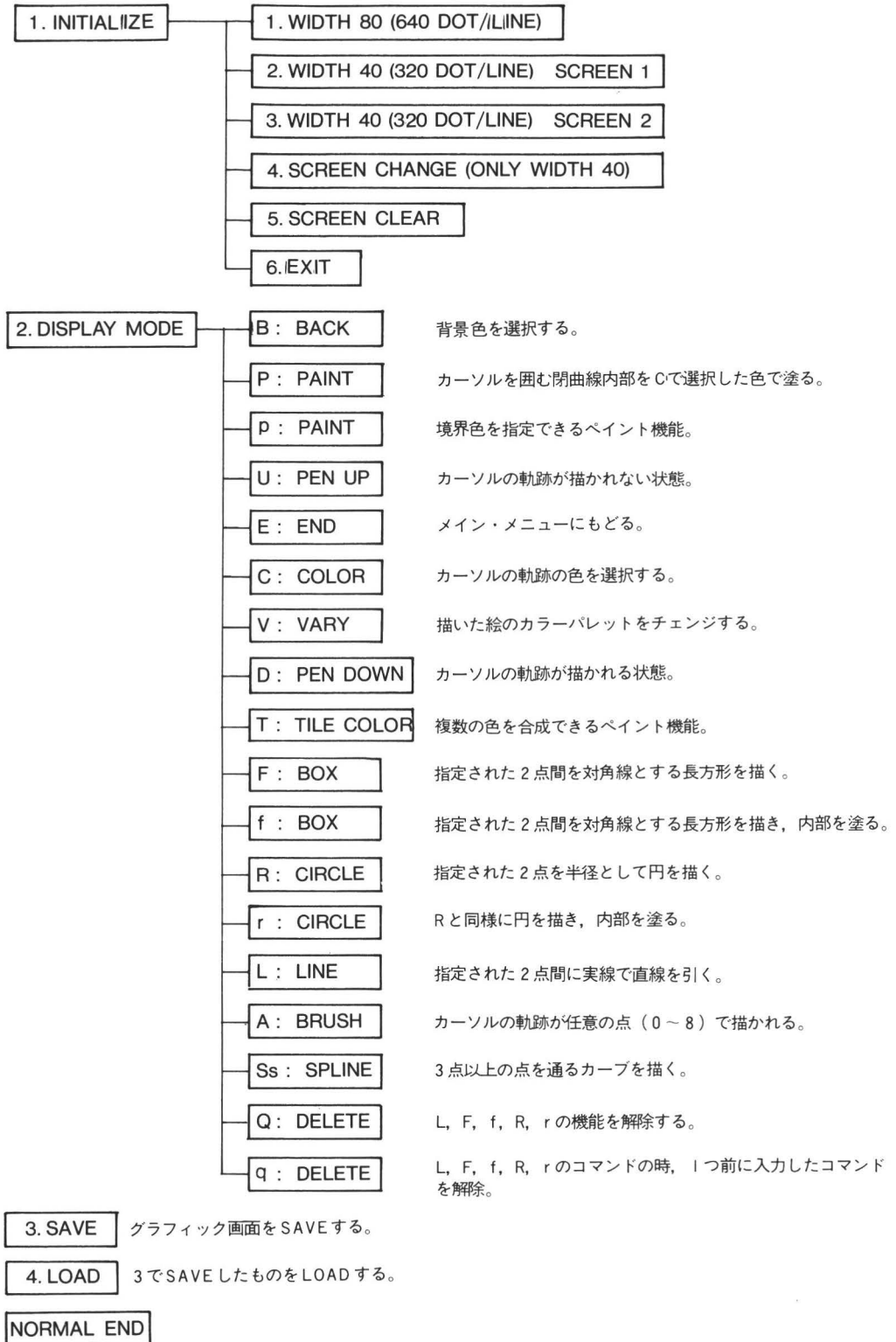



図2―8　LOAD時の画面

デザインツール・プログラムのメニュー構造

デザインツール・プログラムの内容

このプログラムは，大きく画像を作るためのサブルーチンと，それをフロッピー装置やデータレコーダに記憶させたり，呼び出したりするLOAD／SAVEのサブルーチンに分かれます。基本的には，フローチャートや変数表を参考にしていただければ理解できるかと思いますが，簡単に概略を紹介します。

行番号100～210行は，初期設定です。基本的に，コンピュータ・メモリー上にデータは記憶させていないと解説しましたが，160行と170行の配列変数は，スプライン曲線を描くうえでのデータを記憶させておくためのものです。スプライン曲線の点群は，202個まで持つことができます。

行番号300～360行でメインメニュー表示とキー入力を行っています。

400～580行で，描くグラフィック画面の画面設定，600～650行は背景色の設定ルーチン，700～750行で指示ペンのカラー指定，800～1070行でPALET色変換ルーチン，1200～1480行でディスプレイ・コマンドのメニュー表示を行っています。

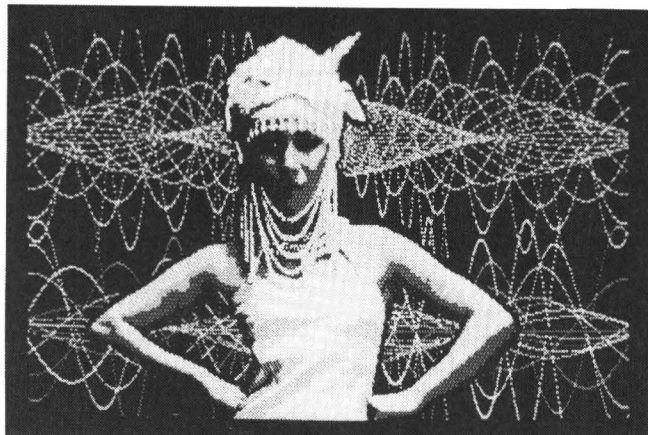
1500～1610行で十字カーソルを読み込んで取り出すルーチン，1620～1830行でコマンド表示と指示メニュー，1840～1970行でジョイステックによる十字カーソル移動量の指示，2000～2190行で十字カーソルの移動のためのルーチン，2200～2250行は直線を引くサブルーチン，2300～2350行でボックスを描き，2500～2590行で円弧を描く，2590行は円弧内を原点から塗りつぶします。

2600～2770行で，描いた図形の削除のために，黒色で塗りつぶします。2800～2870行で乱数により点群を打ち出します。3000～3600行でスプライン曲線を描くためのルーチン，4000～4700行で中間色を作るルーチンを行っています。5000～5170行は画面のLOAD，6000～6280行で画面のSAVE，7000～7100行で入力デバイスの設定，9000～9210行はエラー処理ルーチン，9500～9520行でカラー名のデータを表しています。

いずれのプログラムも，詳しいことは紙面の関係で省略しますが，別著「プログラミング・テクニック集」（学研版）を参考にしていただければ，十字カーソルをどう作ったらいいのか，どう動かしたらいいのか等，詳細に解説してあります。興味のある方は，お読みください。

デザインツール・コマンドのまとめ

コマンド	機　能	行番号	コマンド	機　能	行番号
B	背景色の設定	610～ 650	T	タイリングペイント	4010～4700
C	ペンカラーの設定	710～ 750	V	カラーパレットの変更	810～ 910
E	メインメニュー復帰		／	メニューの切り替え	1060～1070
F	長方形	2310～2350	f	長方形のペイント	2410～2420
L	直線	2210～2250	p	境界線のペイント	930～1050
P	ペイント	920	q	デリート	2610～2770
Q	コマンド解除	2430～2470	r	円のペイント	2560～2590
R	円	2510～2550	s	スプライン実行	3100～3600
S	スプライン曲線の設定	3010～3090			

デザインツール・プログラム 出力例

テクノ・ウーマン

変数とその内容			
配列変数			
PAL（ ）	パレット番号	X（ ）	スプライン点のＸ座標
PC（ ）	境界色	Y（ ）	スプライン点のＹ座標
CS（ ）	カーソルデータ	CP（ ）	スプライン点上の色
CSO（ ）	カーソル上のデータ退避用	CL$（ ）	色の名前
ZB（ ）	スプライン計算用	DM$（ ）	中間色タイリングパターンデータ
ZC（ ）		BL$（ ）	中間色　青色データ
ZB1（ ）		RD$（ ）	中間色　赤色データ
ZB2（ ）		GR$（ ）	中間色　緑色データ
ZB3（ ）			
ZB4（ ）			
変　数			
MY	ディスプレイメニュー表示位置（縦）	AB	ブラシのサイズ
MI	画面のWIDTH	L	ラインの判定
W		R	円の判定
XE	画面横の最大値	B	ボックスの判定
YE	画面縦の最大値	RR	円ペイントの判定
SC	WIDTH 40の画面	BB	ボックスペイントの判定
BC	背景色番号	SP	スプラインの判定
PC	ペンカラー	M	スプラインの点の数
LM	ディスプレイメニュー切り替え	LP	LoopまたはNoon Loopの判定
X	カーソル現在位置のＸ座標	NM$	タイリングパターンの文字列
Y	カーソル現在位置のＹ座標	TL$	タイリングパターン
F	ペンのup, down判別	TL1$	
JST	カーソルキーまたはジョイスティック		

コスモス

デザインツール・プログラムの流れ

初期設定　100～ 210

カーソルキーの選択　7000～7100

画面設定　230

メインルーチン　300～ 360

MN 1 → 画面設定　400～ 580

2 → DISPLAY ルーチン　1500～1970

3 → 画面 SAVE　6000～6280

4 → 画面 LOAD　5000～5170

DISPLAY

初期設定　1500～1530

カーソル表示　1540～1580

キー入力　1590～1600

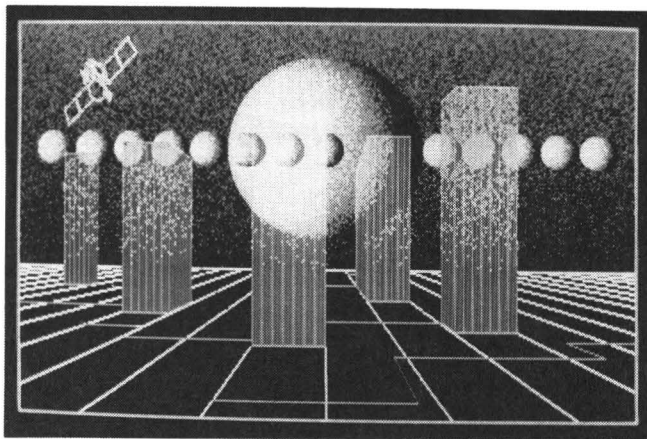
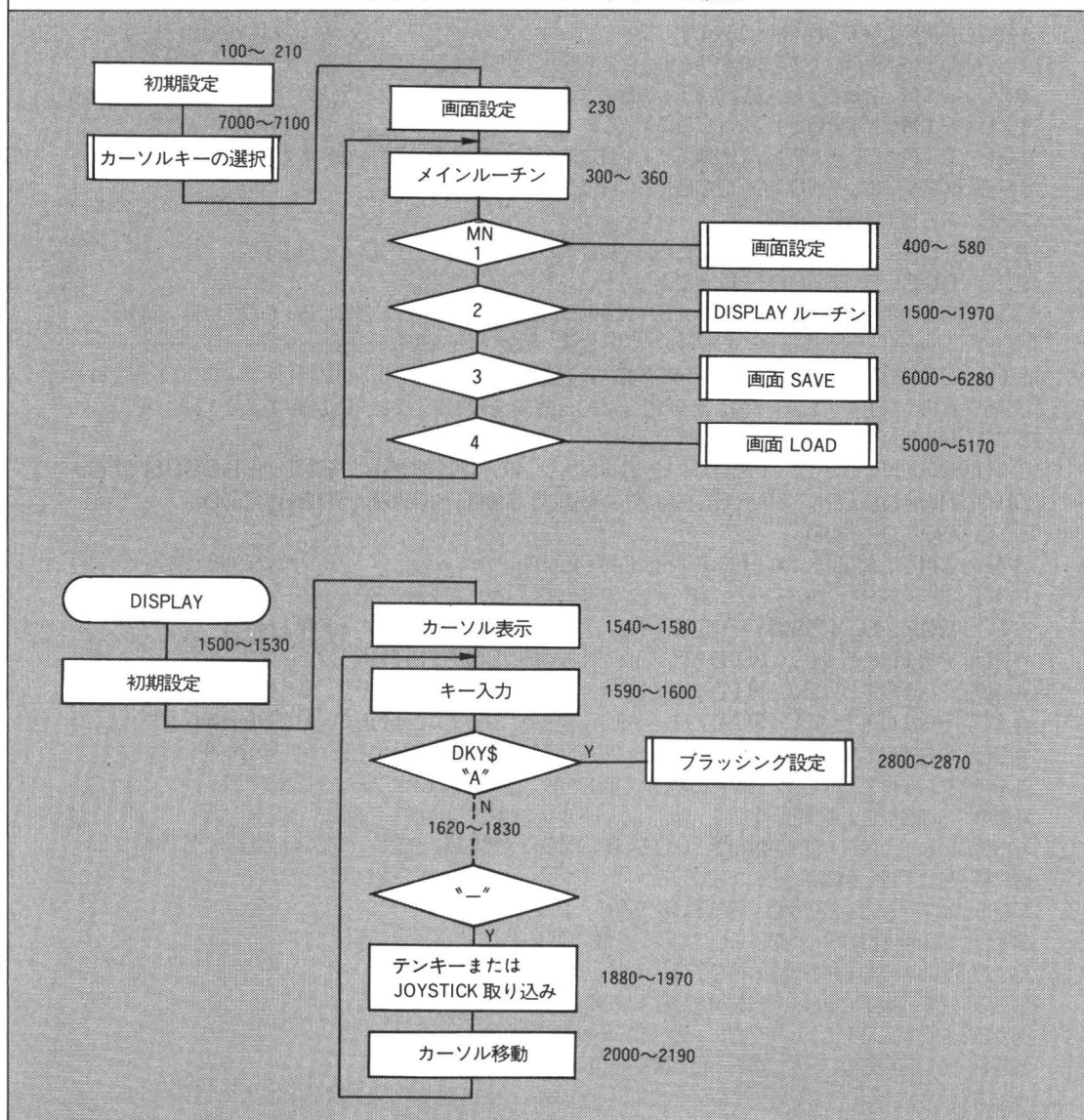
DKY$ "A"　Y → ブラッシング設定　2800～2870

N　1620～1830

"_"

Y

テンキーまたは JOYSTICK 取り込み　1880～1970

カーソル移動　2000～2190

デザインツール・プログラムリスト

```
1 '************************************
2 '*                                  *
3 '*     DESIGN TOOL PROGRAM  V.2     *
4 '*                                  *
5 '*      Programmed By [ SAIMU ]     *
6 '*                                  *
7 '*           1985 . 4 . 1           *
8 '*                                  *
9 '************************************
100 '---------- ｼｮｷ ｾｯﾃｲ ----------------------
110 ON ERROR GOTO 9000
120 WIDTH 40,25,0,1:KLIST 0:KMODE 0    :' (X1 turbo)
130 'WIDTH 40                          :' (X1      )
140 DEFINT A-Y:INIT
150 DIM PAL(7),PC(6),CS(5),CS0(5)
160 DIM ZB(3),ZC(3),ZB1(2),ZB2(2),ZB3(2),ZB4(2)
170 DIM X(202),Y(202),CP(202)
180 DIM CL$(7),DM$(7),BL$(7),RD$(7),GR$(7)
190 KEY 5,"GOTO 200"+CHR$(13)
200 RESTORE 9510
210 FOR I=0 TO 7:READ CL$(I):PAL(I)=I:NEXT
220 GOSUB 7000:MY=22
230 WI=80:W=WI/40:XE=WI*8-1:YE=199:SC=0:GOSUB 500
300 '---------- ﾒｲﾝ ﾌﾟﾛｸﾞﾗﾑ -------------------
310 GOSUB 1390:PRINT"[ DISPLAY TOOL MENU 2 ]":PRINT
320 PRINT" 1. INITIALIZE":PRINT" 2. DISPLAY MODE"
330 PRINT" 3. SAVE"
340 PRINT" 4. LOAD":PRINT" 5. NORMAL END":GOSUB 1450
350 MN=NO:ON MN GOSUB 400,1500,6000,5000,370
360 GOTO 300
370 INIT:CLS 4:WIDTH 80:END
400 '----------- ｶﾞﾒﾝ ｾｯﾃｲ ---------------------
410 GOSUB 1390:PRINT"[ INITIALIZE ]":PRINT
420 PRINT" 1. WIDTH 80 (640 DOT/LINE)"
430 PRINT" 2. WIDTH 40 (320 DOT/LINE) SCREEN 1"
440 PRINT" 3. WIDTH 40 (320 DOT/LINE) SCREEN 2"
450 PRINT" 4. SCREEN CHANGE (ONLY WIDTH 40)"
460 PRINT" 5. SCREEN CLEAR":PRINT" 6. EXIT"
470 GOSUB1450
480 MN1=NO:ON MN1 GOSUB 500,510,520,530,580,570
490 GOTO 400
500 WI=80:SC=0:GOTO 560
510 WI=40:SC=0:GOTO 560
520 WI=40:SC=1:GOTO 560
530 IF WI=80 THEN RETURN
540 SC=SC+1:IF SC=2 THEN SC=0
550 SCREEN SC,SC,0:RETURN
```

```
560 SCREEN SC,SC,0:WIDTH WI:RETURN
570 W=WI/40:XE=WI*8-1:RETURN 300
580 CLS0:RETURN
600 '----------  ハイケイ ショク ノ セッテイ  ---------------
610 GOSUB 1390:GOSUB 1200:GOSUB 1410
620 PRINT"BACK COLOR (8=RETURN) ? ";
630 C=VAL(INKEY$(1)):PRINTC:IF (C>8)+(C<0) THEN 620
640 IF C=8 THEN 1290
650 BC=C:GOSUB 1430:GOTO 620
700 '----------  ペン カラー ノ セッテイ  ----------------
710 GOSUB 1390:GOSUB 1200:GOSUB 1410
720 PRINT"PEN COLOR (8=RETURN) ? ";
730 C=VAL(INKEY$(1)):PRINTC:IF (C>8)+(C<0) THEN 720
740 IF C=8 THEN 1290
750 PC=C:GOSUB 1430:GOTO 1290
800 '----------  パレット カラー ノ ヘンコウ  --------------
810 GOSUB 1390:GOSUB 1200
820 GOSUB 1410:PRINT"COLOR CHANGE FROM ";
830 PC1=VAL(INKEY$(1))
840 IF (PC1<0)+(PC1>8) THEN 820
850 IF PC1=8 THEN 1290
860 CLS:PRINT"COLOR CHANGE FROM ";CL$(PC1);" TO ";
870 PC2=VAL(INKEY$(1))
880 IF (PC2<0)+(PC2>8) THEN 820
890 IF PC2=8 THEN 1290
900 PALET PC1,PC2:PAL(PC1)=PC2:GOSUB 1200:GOTO 820
910 FOR I=0 TO 7:PAL(I)=I:NEXT:PALET:RETURN
920 PAINT (X,Y),PC,1,2,3,4,5,6,7:CC=PC:RETURN
930 GOSUB 1390:GOSUB 1200:GOSUB 1410
940 INPUT"BOUNDARY COLOR (8=RETURN) ? ",CL$
950 IF LEN(CL$)>7 OR LEN(CL$)<1 THEN 940
960 FOR I=1 TO 7
970   IF LEN(CL$)<I THEN PC(I-1)=PC:GOTO 1010
980   A$=MID$(CL$,I,1)
990   IF ASC(A$)<48 OR ASC(A$)>55 THEN 940
1000  PC(I-1)=VAL(A$)
1010 NEXT
1020 PC0=PC(0):PC1=PC(1):PC2=PC(2):PC3=PC(3)
1030 PC4=PC(4):PC5=PC(5):PC6=PC(6)
1040 PAINT(X,Y),PC,PC0,PC1,PC2,PC3,PC4,PC5,PC6
1050 GOSUB 1290:RETURN
1060 LM=-LM:ON (LM+3)/2 GOSUB 1330,1290
1070 RETURN
1200 '----------  ディスプレイ メニュウ ノ ヒョウジ  ---------
1210 GOSUB 1420
1220 FOR I=0 TO 7
1230   LOCATE W*10*(I MOD 4),MY+INT(I/4)
```

```
1240   PRINT USING"#";I;
1250   IF I=0 THEN PRINT" □ ";:GOTO 1270
1260   COLOR I:PRINT" ■ ";:COLOR 7:ELSE PRINT" □ ";
1270   PRINT CL$(PAL(I));:NEXT:RETURN
1280 NEXT:RETURN
1290 GOSUB 1420:PRINT"B: BACK   Pp: PAINT U: PEN ";
1300 PRINT"UP   E: END  ";:IF W=2 THEN PRINT
1310 PRINT"C: COLOR  V: VARY   D: PEN DOWN T:TILE";
1320 IF W=1 THEN 1360
1330 IF W=1 THEN GOSUB 1420 ELSE CONSOLE MY,3,40,40
1340 PRINT"Ff: BOX    Rr: CIRCLE   L: LINE       "
1350 PRINT"A: BRUSH   Ss: SPLINE   Qq: DELETE     ";
1360 GOSUB 1410:CREV 1
1370 PRINT"  ";CL$(PAL(PC));"  ";
1380 CFLASH1:PRINT DKY$;AB;:CFLASH0:CREV0:GOTO 1470
1390 PRW 0:CONSOLE:CLS:GOTO 1430
1400 PRW 0:CONSOLE MY,2:CLS:GOTO 1430
1410 PRW 0:CONSOLE MY+2,1:CLS:GOTO 1430
1420 PRW0:CONSOLE MY,3:CLS
1430 IF BC=7 THEN COLOR 3,BC:RETURN
1440 COLOR 7-BC,BC:RETURN
1450 GOSUB 1410:PRINT"Select Number =";
1460 NO=VAL(INKEY$(1)):PRINT NO:RETURN
1470 LOCATE 15,MY+2:PRINTUSING "###";X;:PRINT",";
1480 PRINTUSING"###";Y;:RETURN
1500 '----------  ﾃﾞｨｽﾌﾟﾚｲ ﾌﾟﾛｸﾞﾗﾑ  ---------------
1510 GOSUB 1390:X=W:Y=1:PC=7:DKY$="U":GOSUB 1290
1520 LM=1:XX=W:YY=1:F=0:R=0:B=0:RR=0:BB=0:SP=0
1530 CC0=0:CC=0:X0=W:Y0=1:XL=W:YL=1:F=0
1540 GET@ (0,0)-(2*W,2),CS,7
1550 LINE (0,0)-(2*W,2),PSET,0,BF
1560 LINE (W,0)-(W,2),PSET,7
1570 LINE (0,1)-(2*W,1),PSET,7
1580 GET@ (0,0)-(2*W,2),CS0,7
1590 IF JST<>0 THEN DKY$=INKEY$:GOTO 1610
1600 DKY$=INKEY$(1)
1610 PUT@ (X-W,Y-1)-(X+W,Y+1),CS,PSET,7
1620 IF DKY$="A" THEN GOSUB 2800:GOSUB 1360
1630 IF DKY$="B" THEN GOSUB 610
1640 IF DKY$="C" THEN GOSUB 700
1650 IF DKY$="D" THEN GOSUB 1360:F=1
1660 IF DKY$="E" THEN GOSUB 2430:GOTO 2460
1670 IF DKY$="F" THEN GOSUB 1360:GOSUB 2300
1680 IF DKY$="L" THEN GOSUB 1360:GOSUB 2200
1690 IF DKY$="P" THEN GOSUB 920
1700 IF DKY$="Q" THEN GOSUB 2430
1710 IF DKY$="R" THEN GOSUB 1360:GOSUB 2500
```

```
1720 IF DKY$="S" THEN GOSUB 3000:GOSUB 1360
1730 IF DKY$="T" THEN GOSUB 4000:GOSUB 1290
1740 IF DKY$="U" THEN GOSUB 1360:F=0
1750 IF DKY$="V" THEN GOSUB 1360:GOSUB 800
1760 IF DKY$="/" THEN IF W=1 THEN GOSUB 1060
1770 IF DKY$="f" THEN GOSUB 2400
1780 IF DKY$="p" THEN GOSUB 930:CC=PC
1790 IF DKY$="q" THEN GOSUB 2600
1800 IF DKY$="r" THEN GOSUB 2560
1810 IF DKY$="s" THEN GOSUB 3100:SP=0
1820 IF DKY$="0" THEN X=W:Y=1      :GOTO 2050
1830 IF DKY$="-" THEN X=WI*4:Y=100:GOTO 2050
1840 GET@ (X-W,Y-1)-(X+W,Y+1),CS,7
1850 PUT@ (X-W,Y-1)-(X+W,Y+1),CS0,PSET,PC
1860 STK=STICK(JST)
1870 PUT@ (X-W,Y-1)-(X+W,Y+1),CS,PSET,7
1880 IF JST<>0 THEN F=0:IF STRIG(JST) THEN F=1
1890 IF STK=1 THEN X=X-W:Y=Y+1:GOTO 2000
1900 IF STK=2 THEN Y=Y+1      :GOTO 2000
1910 IF STK=3 THEN X=X+W:Y=Y+1:GOTO 2000
1920 IF STK=4 THEN X=X-1      :GOTO 2000
1930 IF STK=6 THEN X=X+1      :GOTO 2000
1940 IF STK=7 THEN X=X-W:Y=Y-1:GOTO 2000
1950 IF STK=8 THEN Y=Y-1      :GOTO 2000
1960 IF STK=9 THEN X=X+W:Y=Y-1:GOTO 2000
1970 PUT@ (X-W,Y-1)-(X+W,Y+1),CS0,PSET,PC:GOTO 1590
2000 '---------- ｶｰｿﾙ ﾉ ｲﾄﾞｳ ----------------
2010 IF X>XE-W THEN X=XE-W
2020 IF X<W THEN X=W
2030 IF Y>YE-1 THEN Y=YE-1
2040 IF Y<1 THEN Y=1
2050 IF Y<24 AND MY=0 GOSUB 1420:MY=22:GOTO 2190
2060 IF Y>176 AND MY=22 GOSUB 1420:MY=0:GOTO 2190
2070 GOSUB 1470
2080 IF F=0 THEN 2140
2090 CC=POINT(X,Y):IF AB<>0 GOSUB 2840:GOTO 2110
2100 LINE (XX,YY)-(X,Y),PSET,PC
2110 GET@ (X-W,Y-1)-(X+W,Y+1),CS,7
2120 PUT@ (X-W,Y-1)-(X+W,Y+1),CS0,PSET,PC
2130 GOTO 2180
2140 PUT@ (XX-W,YY-1)-(XX+W,YY+1),CS,PSET,7
2150 CC=POINT(X,Y)
2160 GET@ (X-W,Y-1)-(X+W,Y+1),CS,7
2170 PUT@ (X-W,Y-1)-(X+W,Y+1),CS0,PSET,PC
2180 XX=X:YY=Y:GOTO 1860
2190 GOSUB 1290:GOTO 2080
```

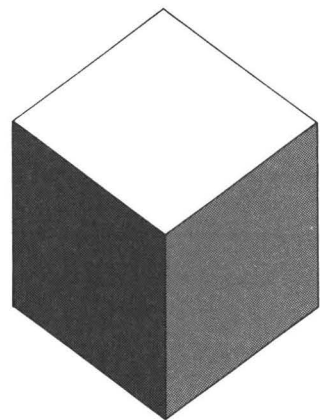
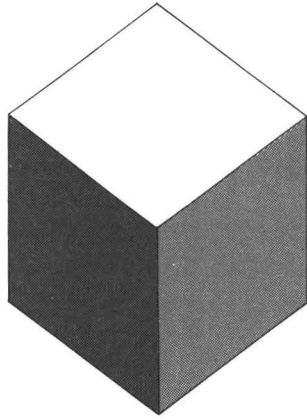
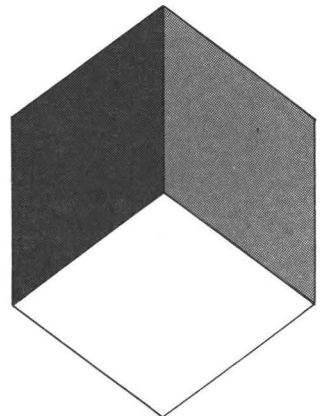
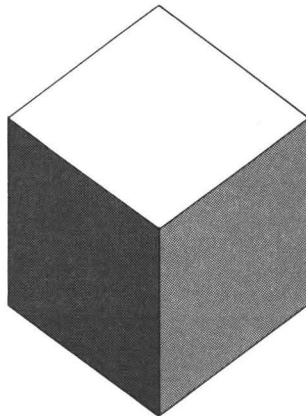
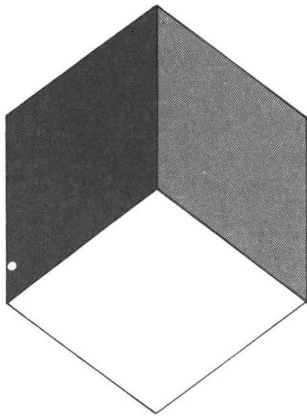
```
2200 '---------- ﾗｲﾝ ----------------------------
2210 IF L=0 THEN GOSUB 2430:L=1:GOTO 2340
2220 LINE (XL,YL)-(X,Y),PSET,PC
2230 IF L=1 THEN CC0=PC
2240 L1=1:X0=XL:Y0=YL
2250 XL=X:YL=Y:CC=PC:PSET (X,Y,CC):RETURN
2300 '---------- ﾎﾞｯｸｽ --------------------------
2310 IF B=0 THEN 2330
2320 LINE (XL,YL)-(X,Y),PSET,PC,B:BB=0:GOTO 2230
2330 GOSUB 2430:B=1
2340 L1=0
2350 X0=X:Y0=Y:CC0=POINT(X,Y):GOTO 2250
2400 '---------- ﾎﾞｯｸｽ ﾍﾟｲﾝﾄ --------------------
2410 IF B=0 THEN 2330
2420 LINE (XL,YL)-(X,Y),PSET,PC,BF:BB=1:GOTO 2230
2430 IF SP=1 THEN GOSUB 3560:SP=0
2440 BB=0:RR=0:L1=0:IF (L=0)*(B=0)*(R=0)THEN RETURN
2450 L=0:B=0:R=0:PSET(X0,Y0,CC0):RETURN
2460 IF F=0 THEN PSET (X,Y,CC) ELSE PSET (X,Y,PC)
2470 RETURN
2500 '---------- ｴﾝ -----------------------------
2510 IF R=0 THEN 2550
2520 RR=CINT(SQR(((X0-X)/W)^2+(Y0-Y)^2))
2530 CIRCLE(X0,Y0),RR,PC,R:RR=0:GOTO 2250
2540 RR=0:GOTO 2250
2550 GOSUB 2430:R=1:GOTO 2350
2560 IF R=0 THEN 2550
2570 RR=CINT(SQR(((X0-X)/W)^2+(Y0-Y)^2))
2580 CIRCLE (X0,Y0),RR,PC,R:PSET(X0,Y0,0)
2590 PAINT (X0,Y0),PC:RR=1:GOTO 2250
2600 '---------- ﾃﾞﾘｰﾄ --------------------------
2610 IF SP=1 THEN GOSUB 3520
2620 IF L=1 THEN 2750
2630 IF BB=1 THEN 2760
2640 IF B=1  THEN 2770
2650 IF RR=1 THEN 2710
2660 IF R=1  THEN 2690
2670 RETURN
2680 X=X0:Y=Y0:XX=X:YY=Y:XL=X:YL=Y:CC=PC:RETURN
2690 RR=CINT(SQR(((X0-XL)/W)^2+(Y0-YL)^2))
2700 CIRCLE (X0,Y0),RR,0,R::GOTO 2740
2710 RR=CINT(SQR(((X0-XL)/W)^2+(Y0-YL)^2))
2720 CIRCLE (X0,Y0),RR,0,R:PSET(X0,Y0,PC)
2730 PAINT(X0,Y0),0:PSET(X0,Y0,CC0)
2740 X=XL:Y=YL:XX=X:YY=Y:CC=0:RETURN
2750 LINE (X0,Y0)-(XL,YL),PSET,0:GOTO 2680
2760 LINE (X0,Y0)-(XL,YL),PSET,0,BF:GOTO 2680
```

```
2770 LINE (X0,Y0)-(XL,YL),PSET,0,B :GOTO 2680
2800 '----------  ｴｱｰ ﾌﾞﾗｼ  --------------------
2810 GOSUB 1410:INPUT "BRUSH SIZE (0 - 8) =",AB
2820 IF AB<0 OR AB>8 THEN 2810
2830 RETURN
2840 FOR I=1 TO AB
2850   BR=AB*RND(1):BG=PAI(2*RND(1))
2860   PSET(2*BR*SIN(BG)+X,BR*COS(BG)+Y,PC)
2870 NEXT:RETURN
3000 '----------  ｽﾌﾟﾗｲﾝ ｶｰﾌﾞ  ------------------
3010 SP=1:IF M=0 THEN 3050
3020 IF M>1 AND X=X(M-1) AND Y=Y(M-1) THEN RETURN
3030 X(M)=X:Y(M)=Y:CP(M)=POINT(X,Y):PSET(X,Y,PC)
3040 CC=PC:M=M+1:IF M>200 THEN 5100 ELSE RETURN
3050 GOSUB 1410:PRINT "Spline Curve";
3060 PRINT" 0 = Non Loop 1 = Loop";
3070 PRINT" Push KEY !!";:C$=INKEY$(1)
3080 IF C$="0" OR C$="1" THEN LP=VAL(C$):M=1:RETURN
3090 GOTO 3060
3100 IF M<3 THEN RETURN
3110 IF LP=1 THEN 3140
3120 X(M)=X(1):Y(M)=Y(1):X(0)=X(M-1):Y(0)=Y(M-1)
3130 M=M-1:GOTO 3160
3140 X(M)=X(1):Y(M)=Y(1):X(M+1)=X(2):Y(M+1)=Y(2)
3150 X(0)=X(M-1):Y(0)=Y(M-1)
3160 X1=X(0):Y1=Y(0):X2=X(1):Y2=Y(1)
3170 X3=X(2):Y3=Y(2):GOSUB 3290
3180 FOR II=1 TO M-1
3190   X1=X(II):Y1=Y(II):X2=X(II+1):Y2=Y(II+1)
3200   X3=X(II+2):Y3=Y(II+2):GOSUB 3290
3210   X1=INT(ZB1(1)):Y1=INT(ZB1(2))
3220   FOR ZT=1 TO ZT2
3230     GOSUB 3410:LINE (X1,Y1)-(G,H),PSET,PC
3240     X1=G:Y1=H
3250   NEXT
3260 NEXT
3270 IF LP=1 THEN LINE -(X(1),Y(1)),PSET,PC
3280 M=0:RETURN
3290 ZT2=SQR((X2-X1)^2+(Y2-Y1)^2)
3300 ZT3=SQR((X3-X2)^2+(Y3-Y2)^2)
3310 ZB(1)=-2*ZT3/3/(ZT2+ZT3):ZB(2)=2/3/(ZT2+ZT3)
3320 ZB(3)=-ZT2/3/(ZT2+ZT3)
3330 P1=X1:P2=X2:P3=X3:GOSUB 3440
3340 ZSX1=ZB(1)*ZC(1)+ZB(2)*ZC(2)+ZB(3)*ZC(3)
3350 P1=Y1:P2=Y2:P3=Y3:GOSUB 3440
3360 ZSY1=ZB(1)*ZC(1)+ZB(2)*ZC(2)+ZB(3)*ZC(3)
3370 P1=X1:P2=X2:ZS1=ZSX:ZS2=ZSX1
```

```
3380 E=1:ZSX=ZSX1:GOSUB 3480
3390 P1=Y1:P2=Y2:ZS1=ZSY:ZS2=ZSY1
3400 E=2:ZSY=ZSY1:GOTO 3480
3410 G=ZB1(1)+ZB2(1)*ZT+ZB3(1)*ZT^2+ZB4(1)*ZT^3
3420 H=ZB1(2)+ZB2(2)*ZT+ZB3(2)*ZT^2+ZB4(2)*ZT^3
3430 RETURN
3440 ZC(1)=1.5*(P2-P1)/ZT2
3450 Z=3*((ZT2^2)*(P3-P2)+(ZT3^2)*(P2-P1))
3460 ZC(2)=Z/ZT2/ZT3
3470 ZC(3)=3/ZT3*(P3-P2):RETURN
3480 ZB1(E)=P1:ZB2(E)=ZS1
3490 ZB3(E)=3*(P2-P1)/(ZT2^2)-2*ZS1/ZT2-ZS2/ZT2
3500 Z=2*(P1-P2)/(ZT2^3)+ZS1/(ZT2^2)+ZS2/(ZT2^2)
3510 ZB4(E)=Z:RETURN
3520 IF M<1 THEN RETURN
3530 IF (X=X(M-1))*(Y=Y(M-1)) THEN 3550
3540 PSET(X(M-1),Y(M-1),CP(M-1)):M=M-1:RETURN
3550 PSET (X(M-1),Y(M-1),CC):M=M-1:RETURN
3560 IF M=0 THEN RETURN
3570 FOR I=M-1 TO 1 STEP -1
3580   PSET (X(I),Y(I),CP(I))
3590 NEXT
3600 M=0:RETURN
4000 '----------  ﾁｭｳｶﾝｼｮｸ ﾍﾟｲﾝﾄ  ----------------
4010 GOSUB 1390:GOSUB 1200:GOSUB 1410
4020 INPUT "ﾁｭｳｶﾝｼｮｸ (2-8ｹ) =",NM$
4030 IF LEN(NM$)<2 OR LEN(NM$)>8 THEN 4020
4040 PT=1:NM1$="":TL1$="":DM$="00"
4050 FOR I=1 TO LEN(NM$)
4060   A$=MID$(NM$,I,1)
4070   IF ASC(A$)<48 OR ASC(A$)>55 THEN RETURN
4080   NM1$=NM1$+MID$(NM$,LEN(NM$)-I+1,1)
4090 NEXT
4100 NM$=LEFT$(STRING$(4,NM$),8)
4110 NM1$=LEFT$(STRING$(4,NM1$),8)
4120 FOR I=1 TO 8
4130   NM$(I-1)=MID$(NM$,I,1)+MID$(NM1$,I,1)
4140 NEXT
4150 BL=0:BL1=0:RD=0:RD1=0:GR=0:GR1=0
4160 FOR I=1 TO 8
4170   T=VAL(MID$(NM$,I,1)):T1=VAL(MID$(NM1$,I,1))
4180   GR=GR*2+(T AND 4)/4:GR1=GR1*2+(T1 AND 4)/4
4190   RD=RD*2+(T AND 2)/2:RD1=RD1*2+(T1 AND 2)/2
4200   BL=BL*2+(T AND 1):BL1=BL1*2+(T1 AND 1)
4210 NEXT
4220 BL$ =RIGHT$(DM$+HEX$(BL),2)
4230 BL1$=RIGHT$(DM$+HEX$(BL1),2)
```

```
4240 RD$=RIGHT$(DM$+HEX$(RD),2)
4250 RD1$=RIGHT$(DM$+HEX$(RD1),2)
4260 GR$=RIGHT$(DM$+HEX$(GR),2)
4270 GR1$=RIGHT$(DM$+HEX$(GR1),2)
4280 TL$=BL$+RD$+GR$+BL1$+RD1$+GR1$
4290 FOR J=0 TO 7
4300   BL=0:RD=0:GR=0
4310   FOR I=1 TO 2
4320     T=VAL(MID$(NM$(J),I,1))
4330     GR=GR*2+(T AND 4)/4
4340     RD=RD*2+(T AND 2)/2
4350     BL=BL*2+(T AND 1)
4360   NEXT
4370   BL=BL*85:RD=RD*85:GR=GR*85
4380   BL$(J)=RIGHT$(DM$+HEX$(BL),2)
4390   RD$(J)=RIGHT$(DM$+HEX$(RD),2)
4400   GR$(J)=RIGHT$(DM$+HEX$(GR),2)
4410   TL1$=TL1$+BL$(J)+RD$(J)+GR$(J)
4420 NEXT
4430 CLS:CGEN 1:BL0$="":RD0$="":GR0$=""
4440 FOR I=1 TO 4
4450   BL0$=BL0$+BL$+BL1$:RD0$=RD0$+RD$+RD1$
4460   GR0$=GR0$+GR$+GR1$
4470 NEXT
4480 BL2$="":RD2$="":GR2$=""
4490 FOR I=0 TO 7
4500   BL2$=BL2$+BL$(I)
4510   RD2$=RD2$+RD$(I)
4520   GR2$=GR2$+GR$(I)
4530 NEXT
4540 A$=HEXCHR$(BL0$)+HEXCHR$(RD0$)+HEXCHR$(GR0$)
4550 DEF CHR$(30)=A$
4560 A$=HEXCHR$(BL2$)+HEXCHR$(RD2$)+HEXCHR$(GR2$)
4570 DEF CHR$(31)=A$
4580 CGEN 1:PRINT:PRINT #0,CHR$(30);CHR$(30);
4590 PRINT #0,CHR$(30);CHR$(30);:CGEN 0:GOTO 4620
4600 CGEN 1:PRINT:PRINT #0,CHR$(31);CHR$(31);
4610 PRINT #0,CHR$(31);CHR$(31);:CGEN 0
4620 PRINT " OK ? (Y / N) ";CHR$(5);
4630 TK$=INKEY$(1):IF TK$="Y" THEN 4660
4640 IF TK$="N" THEN 4000
4650 PT=-PT:IF PT>0 THEN 4580 ELSE 4600
4660 IF PT<0 THEN 4690
4670 PAINT (X,Y),HEXCHR$(TL$),1,2,3,4,5,6,7
4680 CC=POINT(X,Y):RETURN
4690 PAINT (X,Y),HEXCHR$(TL1$),1,2,3,4,5,6,7
4700 CC=POINT(X,Y):RETURN
```

```
5000 '---------- ｶﾞﾒﾝ ﾉ ﾛｰﾄﾞ  -------------------
5010 GOSUB 1420:PRINT"[ LOAD ]"
5020 INPUT"  FILE NAME ? ('/'=RETURN) ",T$
5030 IF LEN(T$)>16 THEN 5000
5040 IF T$="/" THEN RETURN
5050 GOSUB 5090
5060 FOR I=0 TO 7
5070   PALET I,PAL(I)
5080 NEXT:RETURN
5090 OPTION SCREEN 2:PRW &HFF
5100 OPEN"I",1,T$:REC=0:GOSUB 6140
5110 IF (WI=40)*(SC=1) THEN REC=4
5120 A$=INPUT$(128,1):B$=INPUT$(128,1)
5130 DEVO$"MEM:",REC,A$,B$:REC=REC+1
5140 IF WI=80 THEN 5160
5150 IF (REC MOD 8)=4*(1-SC) THEN REC=REC+4
5160 IF REC<&HC0 THEN 5120
5170 CLOSE #1:PRW 0:INIT:OPTION SCREEN 1:CLS:RETURN
6000 '---------- ｶﾞﾒﾝ ﾉ ｾｰﾌﾞ  -------------------
6010 PRW0:CONSOLE 0,22:CLS:GOSUB 1430:GOSUB 1390
6020 GOSUB 1420:PRINT"[ SAVE ]"
6030 INPUT"  FILE NAME ? ('/'=RETURN) ",T$
6040 IF LEN(T$)>16 THEN 6000
6050 IF T$="/" THEN RETURN
6060 OPTION SCREEN 2:PRW &HFF
6070 OPEN"O",1,T$:REC=0:GOSUB 6250
6080 IF (WI=40)*(SC=1) THEN REC=4
6090 DEVI$"MEM:",REC,A$,B$:PRINT#1,A$;B$;:REC=REC+1
6100 IF WI=80 THEN 6120
6110 IF (REC MOD 8)=4*(1-SC) THEN REC=REC+4
6120 IFREC<&HC0 THEN 6090
6130 GOTO 5170
6140 INPUT #1,WI2
6150 IF WI2=WI THEN 6170
6160 GOSUB 5170:GOTO 6200
6170 FOR I=0 TO 7
6180   INPUT #1,PAL(I)
6190 NEXT:RETURN
6200 GOSUB 1420
6210 PRINT"WIDTH MISS MATCH !! (Push Any key) ";
6220 KY$=INKEY$(1)
6230 IF KY$="" THEN 6200
6240 RETURN 5000
6250 PRINT #1,WI
6260 FOR I=0 TO 7
6270   PRINT #1,PAL(I)
6280 NEXT:RETURN
```

```
7000 '----------- カーソル キー ノ センタク -----------------
7010 CLS4:WIDTH40:CSIZE 2
7020 COLOR 5:LOCATE 4,4 :PRINT #0 "CHOOSE NUMBER !!"
7030 COLOR 3:LOCATE 6,8 :PRINT #0 "1. KEY BOARD";
7040 COLOR 4:LOCATE 6,12:PRINT #0 "2. JOYSTICK 1";
7050 COLOR 6:LOCATE 6,16:PRINT #0 "3. JOYSTICK 2";
7060 CSIZE 0:CFLASH 1
7070 LOCATE 4,20:INPUT "Push Any Key !! ";JST$
7080 IF (JST$="1")+(JST$="2")+(JST$="3") THEN 7100
7090 GOTO 7070
7100 JST=VAL(JST$)-1:CFLASH 0:RETURN
9000 '----------- エラー ショリ ------------------------
9010 CONSOLE:PRW0
9020 IF (ERR=53)*(ERL=5100) THEN 9100
9030 IF (ERR=53)*(ERL=6070) THEN 9110
9040 IF (ERR=72)*(ERL=6070) THEN 9120
9050 IF (ERR=73)*(ERL=6070) THEN 9130
9060 IF ERR=73 THEN GOSUB 9210:RESUME
9070 GOSUB 5170:PRINT ERR,ERL
9080 PRINT"If you want to go on, Push 'F5' Key"
9090 STOP
9100 GOSUB 9140:RESUME 5000
9110 GOSUB 9140:RESUME 6000
9120 GOSUB 9200:RESUME 6000
9130 GOSUB 9210:RESUME 6000
9140 GOSUB 1390:FILES
9150 PRINT:PRINT"Push Any Key !! ('SP'=FILES) ";
9160 KY$=INKEY$(1)
9170 IFKY$=" "THEN PRINT:FILES:GOTO9150
9180 IF KY$=""THEN 9150
9190 RETURN
9200 GOSUB 1390:PRINT"Write Protected !!":GOTO 9150
9210 GOSUB 1390:PRINT"Device Offline !!" :GOTO 9150
9500 '----------- イロ ノ ナマエ ノ データ -----------------
9510 DATA BLACK,BLUE,RED,MAGENTA
9520 DATA GREEN,CYAN,YELLOW,WHITE
```

3次元図形処理技術入門

3次元デザインツールの製作

3次元図形の表現方法には,「面」「線」それに「曲」「平」の4通りの方法が存在します(表3—1)。これらの中で,どのデータ構造を選択するかは,その出力装置の方法によって異なります。マッピングやレイトレーシング等を行う場合は,図形の面データを必要としますが,単純にXYプロッタ等のペンで出力する場合は,線データのほうがよいといえます。

ここでは,3次元図形をワイヤーフレーム(一部サーフェイス・モデル)で表現します。

データ構造

物体を記述する方法の最も簡単なものは,座標データのみを順番に格納して描かせるという,単純配列型データ構造といえます。この方法は,データ構造が簡単で,必要とする記憶容量も少なくてすみ,理解も早いのですが,一度入れたデータの削除や挿入に時間を要するという欠点があります。

ここでは,階層配列型のデータ構造をとることにします。これは,立体は平面によって構成され,平面は陵線によって囲まれ,陵線は点によって与えられているという構造です(図3—1)。第1章の形状モデリングで解説した直方体を眺めてよく理解してください。

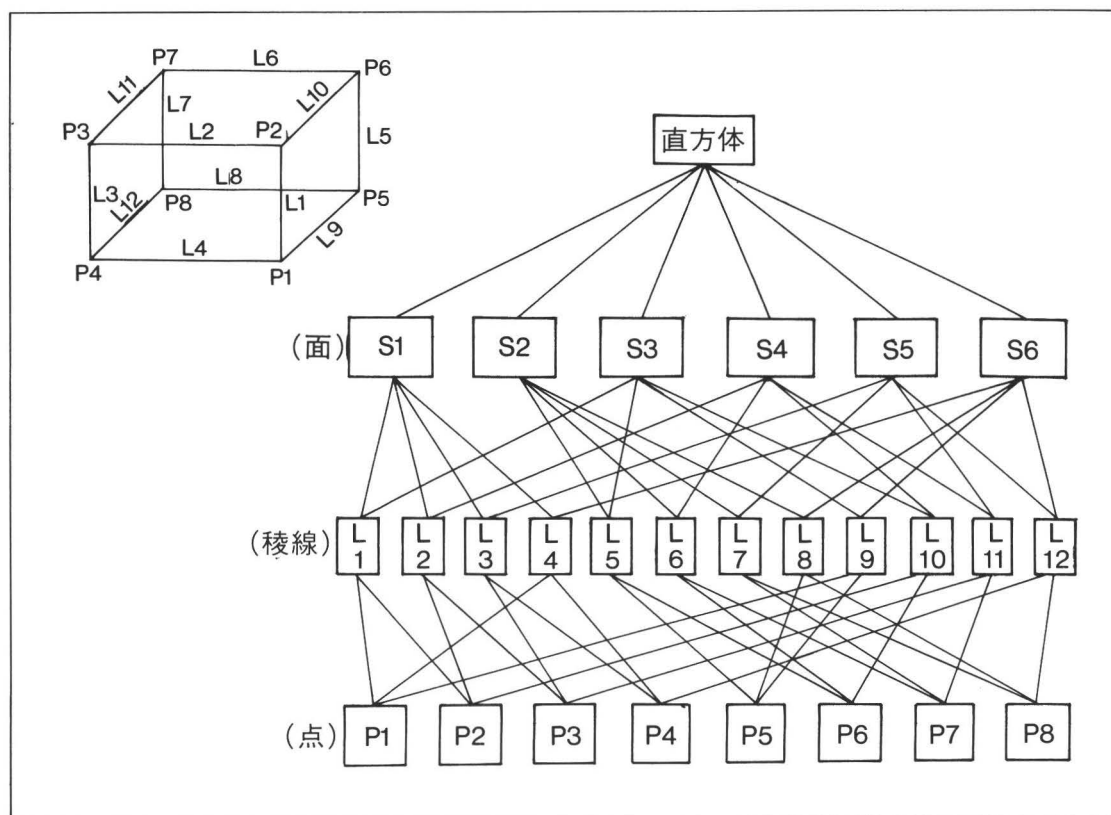



図3—1　3次元図形の表示方法(階層配列型データ構造)

3次元デザインツール・プログラム

このプログラムを入力して実行（RUN）すると，図3—2のように，画面上にメインメニュー表示と，そのコマンド表示，それに，図形を記録しているワールド座標系のスクリーン座標から見た座標軸を表示するワールド座標軸表示が現れます。メインメニュー表示に従ってそれぞれのコマンドを説明します。

PRODUCT

PRODUCT 命令は，ワイヤーフレーム・モデル（一部サーフェイス・モデルのデータ構造をしている）を作成するためのコマンド群です。

```
[ 3D-TOOL MAIN MENU ]

 1. PRODUCT
 2. DISPLAY
 3. LOAD
 4. SAVE
 5. NORMAL END
```

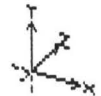
ワールド座標軸表示

```
Push Key (1 - 5) ▌
```

図3－2
メイン・
メニュー表示

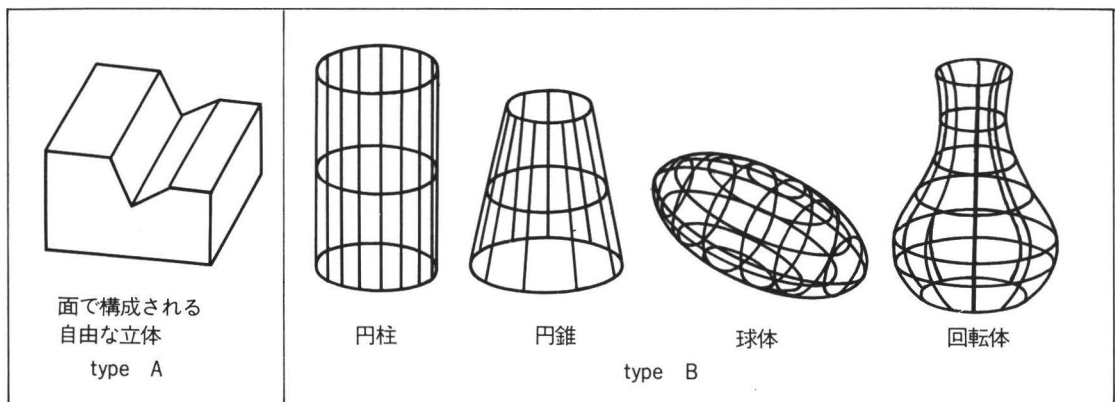



図3－3　3次元システムの図形記述モデルの種類

実際に描くことのできる図形は大別して，面で構成される自由立体（type A）と，円柱，円錐，球体，回転体といった登録された形状記述の行える図形（type B）の2つに分かれています（図3—3)。

type A, Bそれぞれによって入力手順が2種に分かれます。

メインメニューから，1. PRODUCT を選びます（1をキー入力します)。するとサブメニューとして，

```
1. POINT
2. PLANE
3. MANY  P.
4. TRANSFORMATION
5. EXIT
```

が表示されます。これから先はそれぞれのサブメニュー命令を解説しながら，図形記述を行っていきます。先のデータ構造での説明のように，この3次元図形処理プログラムは，サーフェイス・モデルのデータ構造を持たせています。その内容記述の構造に関しては解説済みですので，ここでは実際に図形を記述する方法について解説します。

POINT

POINTは，type Aのような自由立体を作成する場合のPOINT（点）データをコンピュータメモリー上に記憶させるためのコマンドです。1辺が100の立方体を作成する方法について解説を進めます。立方体の角の各点の座標値は，ワールド座標系の原点に図3—4のように記述すると，それぞれ表3—2のような座標値を得ることができます。サブメニュー表示から1をキー入力すると，

```
〔POINT〕
<POINT  No.
```

と，点の番号を聞いてくるので，順次，1CRと入力すると，サブメニュー表示欄に，

```
<POINT  No. 1>
X=  0
Y=  0
Z=  0
```

と表示され，ポイント①のX, Y, Zの座標値を聞いてきます。図3—4の表からポイント①は，Xが0，Yが0，Zが0ですから，そのまま0CR，0CR，0CRと入力します。

次に，ポイント②の座標値は，（100, 0, 0）ですから，2CRとして，

```
<POINT No. 2>
X=  0
Y=  0
Z=  0
```

立方体を構成する点の座標値

	X座標	Y座標	Z座標
①	0	0	0
②	100	0	0
③	100	100	0
④	0	100	0
⑤	0	0	100
⑥	100	0	100
⑦	100	100	100
⑧	0	100	100

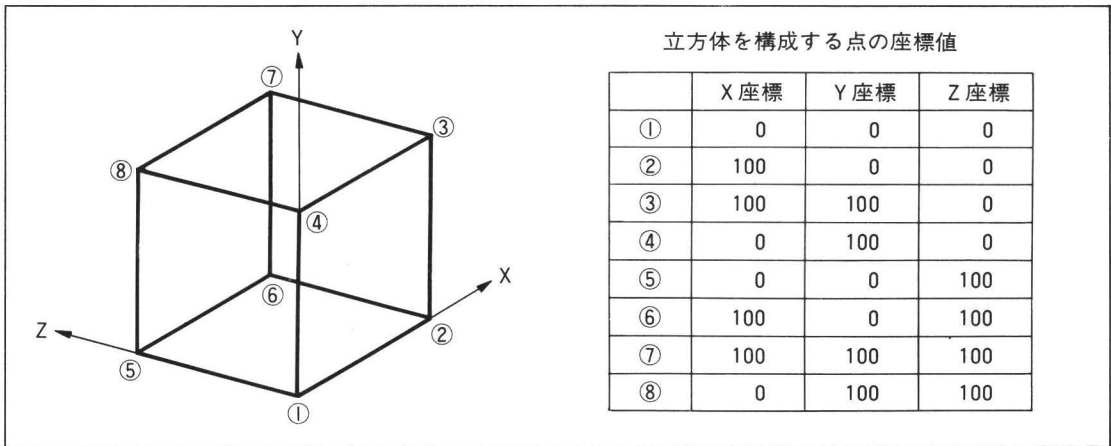



図3－4　ワールド座標系での点によって記述した立方体

のそれぞれの座標値を，[1][0][0][CR]，[0][CR]，[0][CR]と入力してください。以下，このようにして，ポイント③から⑧まで入力を終えたら，[E][CR]を入力してください。これでコンピュータ・メモリー上に，ポイント座標値が記憶されたことになります。画面表示は，サブメニューに戻ります。次いで，PLANEの座標記述に移ります。

PLANE

PLANEは，POINT命令で登録したデータから，上位階層データ構造として面データを作成するためのものです。PLANEは，変数BHPとしてDIMにあるように，50ポイントまでの点群によって記述することが可能です。先ほどの1辺が100の立方体を例にとって，PLANE番号をわかりやすくするために図3—6のように決めておくと，それぞれの面を構成するポイント・ナンバーとして同図の表が得られます。

これらの面データを入力していくことにします。〔PRODUCT〕のサブメニュー表示から[2]を選ぶと，

```
〔PLANE〕
<PLANE  No.
```

と，PLANE（面）の名前を聞いてきます。表に従って[1][CR]を入力すると，

図3－5　PLANE記述例

立体を構成するそれぞれの面を構成する点の番号

面の名前	構成される点の番号			
❶ の 面	①	②	③	④
❷ の 面	⑧	⑤	①	④
❸ の 面	⑤	⑧	⑦	⑥
❹ の 面	⑥	⑦	③	②
❺ の 面	③	④	⑧	⑦
❻ の 面	①	②	⑥	⑤

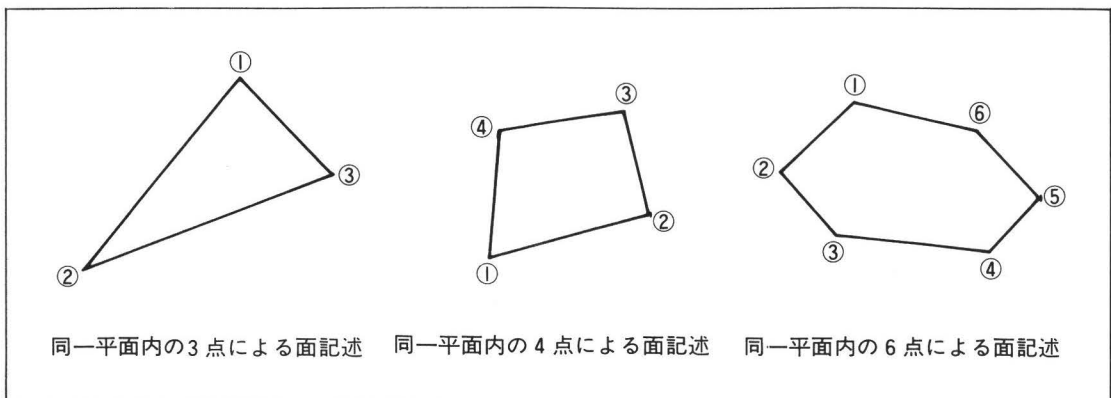
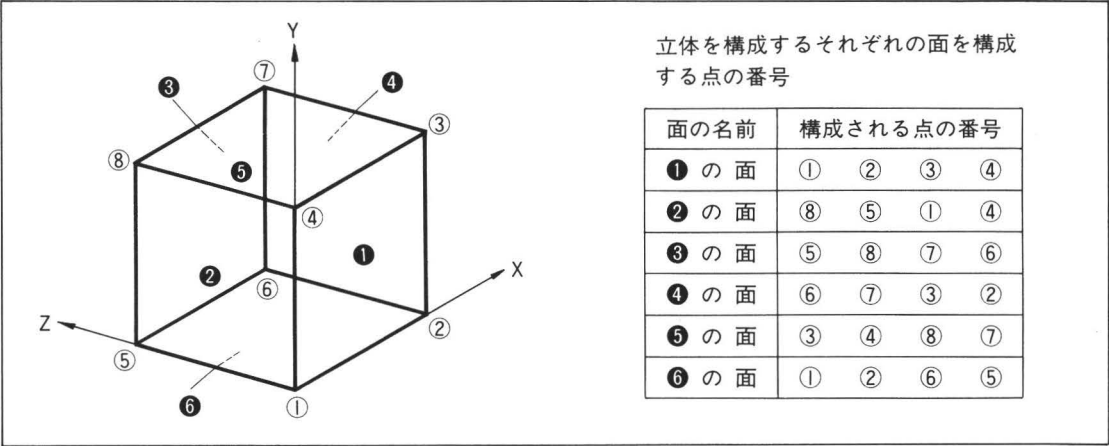



図3－6　立方体を構成する面

```
<PLANE　No. 1>
(1)
Point　number　(0=RETURN)　=　0
```

と聞いてきます。❶の面を構成するポイント番号は，①，②，③，④ですから，1 CR，2 CR，3 CR，4 CRを入力します。そして最後に，そのまま（0表示されている）CRを入力すると，スクリーン座標上に，PLANE ①の四角形が表示され，〈PLANE No.〉上でカーソルが点滅している状態になります。同様にして，PLANE ❷，❸，❹，❺，❻の各面のポイント番号を入力します。全部の入力が終えたら，E CRを入力して，〔PRODUCT〕のサブメニュー表示に戻ってください。

これで，立方体の PLANE 情報の入力は完了したわけです。いよいよ，サーフェイス・モデルの記述を完了するために，〈3.　MANY　P〉を選択します。

MANY P

MANY　P命令は，PLANE命令で記述した面情報を，立体として組み立てるためのものです。現在まで記述している立方体の，〈MANY　P　No.〉を1とすると，この物体は，面❶，❷，❸，❹，❺および❻で構成されています。今までのものと同様に，

```
<MANY　P　No.
```

と表示されている状態で1 CRを入力すると，

```
(1)
Plane　number　(0=RETURN)　=　0
```

と表示されます。ここで，1 CRと入力すると，画面上に PLANE ❶が表示され，

```
(2)
Plane　number　(0=RETURN)　=　0
```

のように変わります。

次いで[2][CR], [3][CR], [4][CR], [5][CR], [6][CR]と順次入力した後, [0][CR]で〈MANY　P No.〉入力のところに戻り, [E][CR]を入力すると, type A の例としての1辺が100の立方体の入力が完了したことになるわけです。

TRANSFORMATION

TRANSFORMATION は, 作りあげた3次元物体を, 実際にプログラム上（ワールド座標系上）に登録するためのものです。登録する物体として, 変数ATRで10個の配列をとっています。〔PRODUCT〕のサブメニュー表示から[4]を選択すると,

```
〔TRANSFORMATION〕
〈TRANSFORMATION  No.
```

と聞いてくるので, 登録 No.（10個まで）を入力します。ここで, この場合は最初の（1辺が100の立方体として）[1][CR]を入力すると,

```
〔TRANSFORMATION〕
1.  MANY  P
2.  SPHERE
3.  CYLINDER
4.  CONE
5.  ROTATION
6.  DELETE
7.  EXIT
```

と表示されて入力待ちとなります。

1. MANY　P

type A の代表である1辺が100の立方体を, ワールド座標系に登録してみます。この場合は, 1. MANY　P ですから, [1]を入力すると,

```
〔TRANS  MANY  P〕
MANY  P  NUMBER  =
```

と聞いてきます。前項〈3. MANY　P〉でこの立方体を[1]として登録しているので, [1][CR]を入力します。

画面表示は次に,

```
COLOR(1－7)  =
```

と聞いてきます。ここで, 表示したい色番号（1=青, 2=赤, 3=シアン, 4=グリーン, 5=シアン, 6=イエロー, 7=白）を入力すると, 次のように聞いてきます。

```
TRANS  SCALE  X  =
```

これは, すでに記述した立方体のうち, X方向のスケールを何倍に書き直せばいいのかを尋ねてい

るもので，従来どおりのスケールだと[1][CR]，それよりも倍率を大きく表示したければ1以上の数値を，縮小するには1以下の数値を入力してください。

X方向のスケールの入力が終わると，

```
TRANS SCALE Y =
```

続いて，

```
TRANS SCALE Z =
```

と聞いてくるので，それぞれY方向とZ方向のスケールを何倍かに指定してください。

例えば，TRANS SCALEの値を，X＝2，Y＝0.5，Z＝1.5とすると，1辺が100の立方体は，図3－7のように，200（X辺の長さ）×50（Y辺の長さ）×150（Z辺の長さ）となります。

記述図形の変換後の図形が決まると，次のように聞いてきます。

```
TRANS ROTATE X =
```

これは，X軸のまわりに何度回転するかを入力するもので，回転の向きは，図3―8のようにX軸の前進する方向に対して右ネジの回転を正とします。同様にして，Y軸およびZ軸に対しても入力します。

この例では，例えば，X＝30，Y＝20，Z＝0とすると，図3－9のように，X，Y，Z軸にそれぞれ30度，20度，30度回転して得られた図形が記述されることとなります。

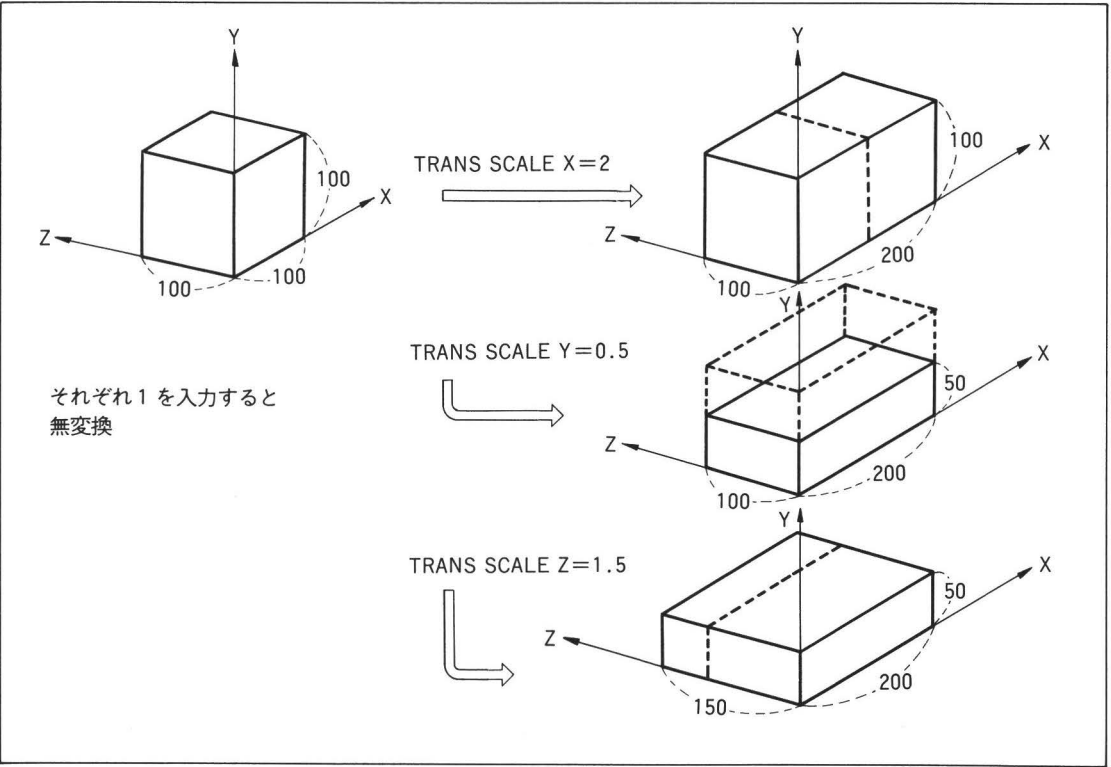



図3－7　TRANS　SCALEによる図形変換の例

次いで,

```
TRANS  MOVE  X  =
```

と，聞いてきます。これは，指定した位置（原点（0,　0,　0））からのX方向の移動量を入力するもので，Y軸，Z軸に関しても同様の指示をします。

これらの指示を完了すると，ワールド座標系に実際に図形を記述したことになります。同様のMANY　Pを，別のTRANSで記述することが可能です（最大10個まで指定できます)。

1辺が100の立方体を例にとって，面で構成される自由立体（type A）についての記述方式について話を進めてきました。TRANSによるワールド座標系への図形記述は，このほかにワイヤーフレ

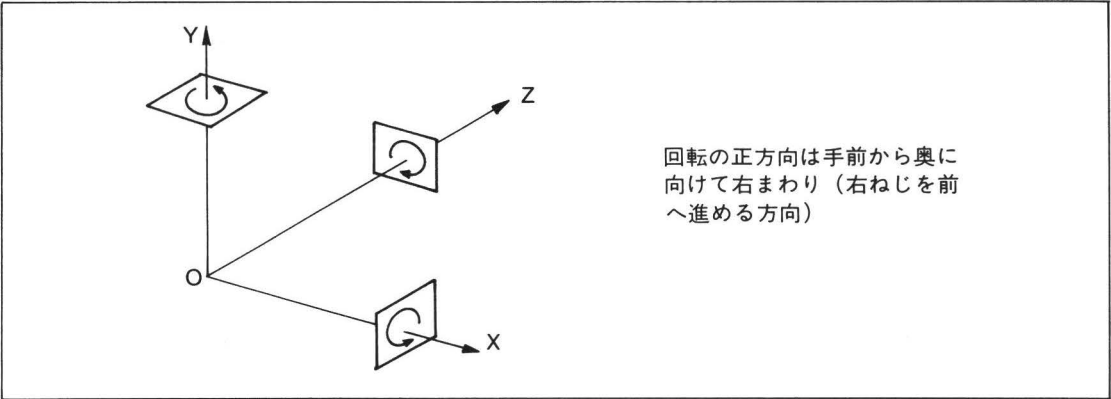



図3－8　回転の正方向

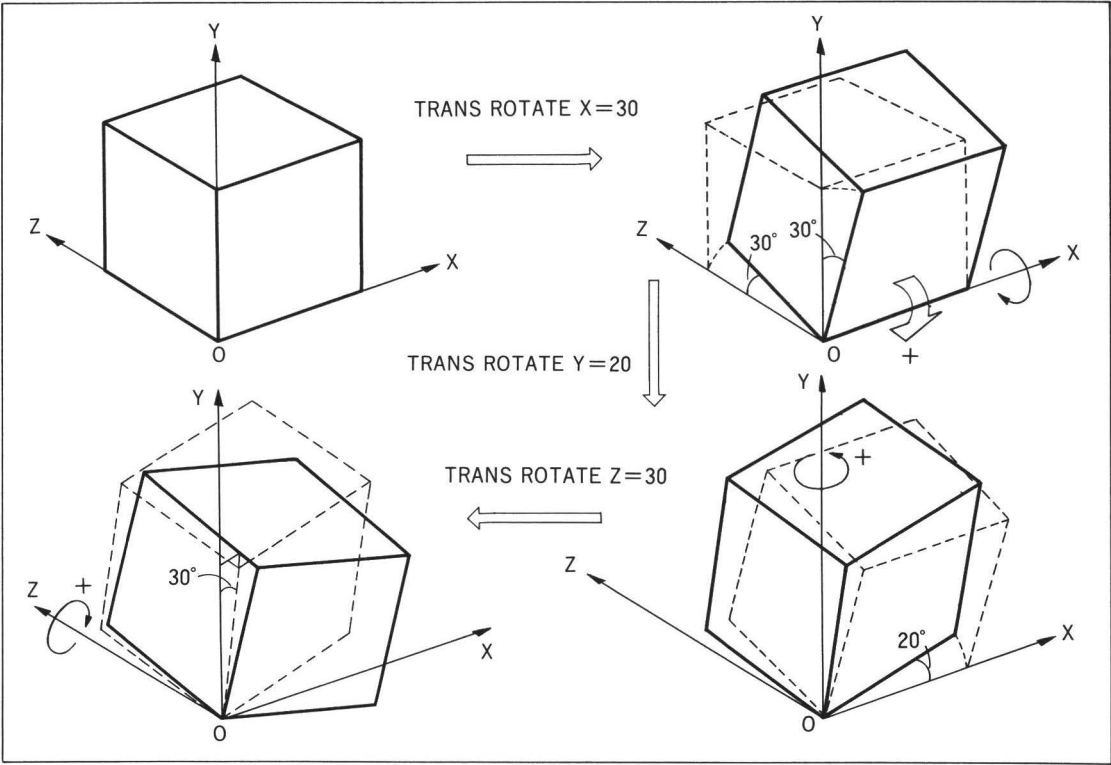



図3－9　TRANS　ROTATEの例

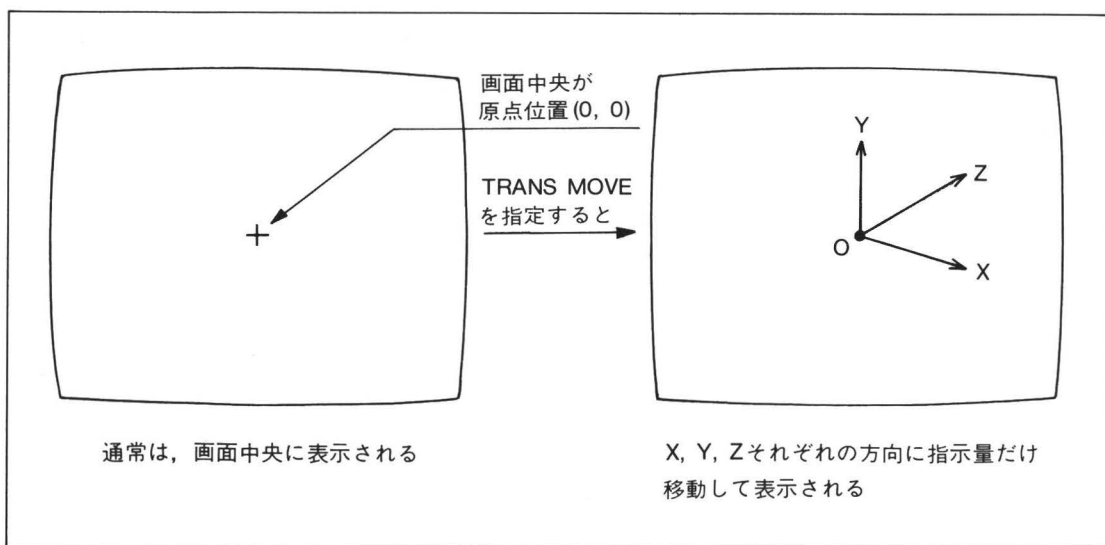



図3—10　スクリーン画面の原点座標軸

ーム・モデルのデータ構造を持つSPHERE（球体），CYLINDER（円柱），CONE（円錐），ROTATION（回転体）に代表される type B（69ページ，図3—3）についても行うことができます。

2．SPHEER（球体）

〔TRANSFORMATION〕のメニューで[2]を入力すると，球体を記述することができます。球体の場合は，

```
RADIUS =
```

と聞いてくるので，球体の半径を入力してください。

次いで，

```
COLOR(1～7) =
TRANS SCALE X =
TRANS SCALE Y =
TRANS SCALE Z =
TRANS ROTATE X =
TRANS ROTATE Y =
TRANS ROTATE Z =
TRANS MOVE X =
TRANS MOVE Y =
TRANS MOVE Z =
```

と聞いてくるので，〈1．MANY　P〉の場合と同様の考え方で，必要なデータを入力してください。

ただし，このうち，TRANS SCALE のX，Y，Zにそれぞれ異なる倍率を入力すると，球体がラグビーボール状に変形するので注意してください。

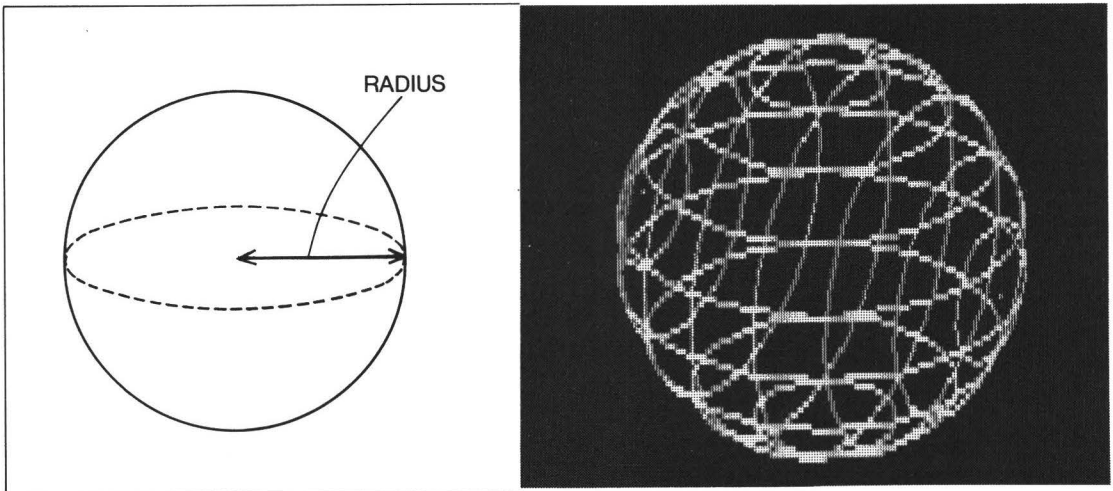



図3—11　球体表現の数値

3. CYLINDER（円柱）

〔TRANSFORMATION〕メニューで[3]を入力すると，円柱を記述することができます。

最初に，

RADIUS =

と聞いてくるので，円柱の半径を入力してください。

次いで，

HEIGHT =

と聞いてくるので，高さを入力してください。

あとは，他と同様に，COLOR, TRANS SCALE, TRANS ROTATE, TRANS MOVE の順でそれぞれの値を入力すれば，目的の円柱を記述することができます。

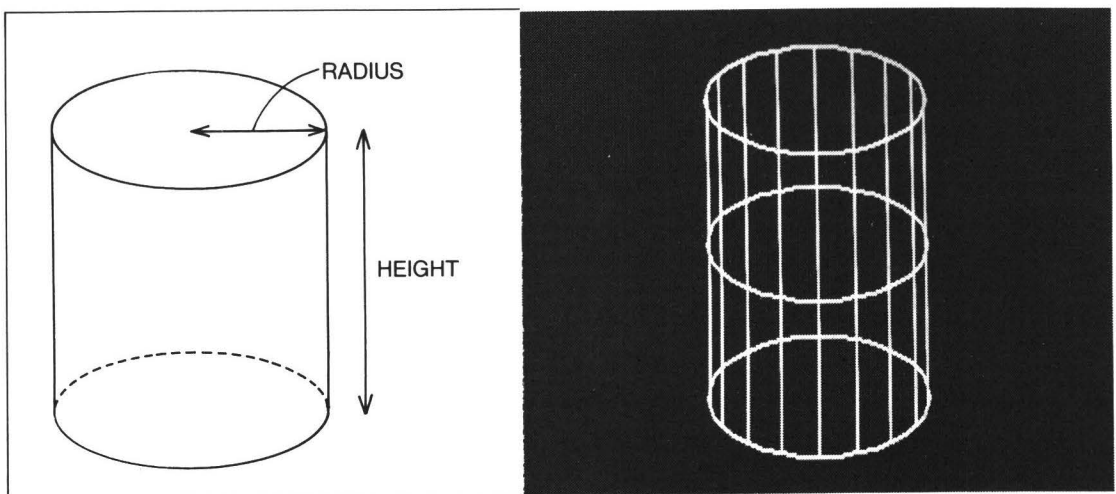



図3—12　円柱表現の数値

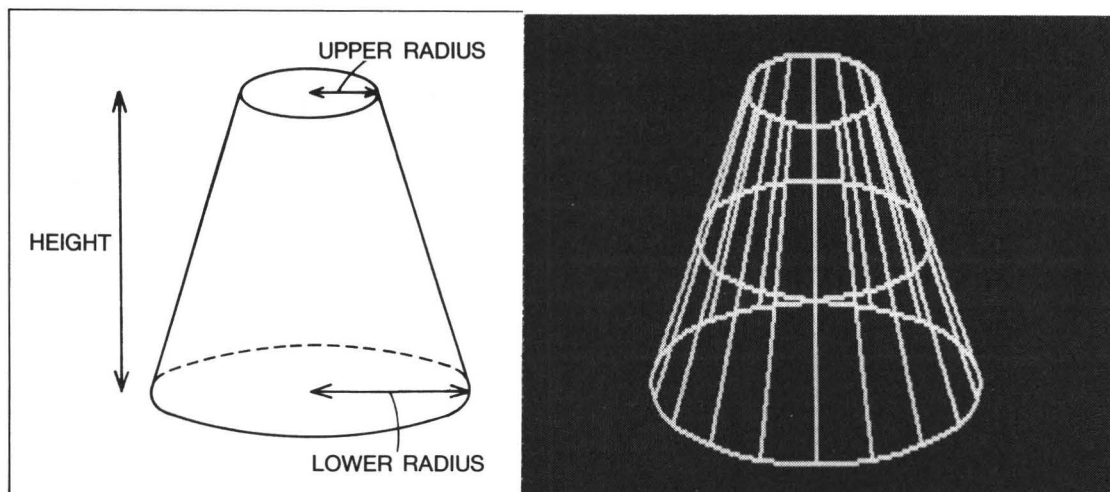



図3―13　円錐台表現の数値

4. CONE (円錐)

〔TRANSFORMATION〕メニューで4を入力すると，円錐を記述することができます。

円錐の場合も，最初に基本形のデータ入力を求めて，

```
UPPER RADIUS =
```

と聞いてくるので，円錐の上底の半径を入力してください。上底は原点の高さになります。さらに，

```
LOWER RADIUS =
```

と聞いてくるので，円錐の下底の半径を入力してください。

次いで，

```
HEIGHT =
```

と聞いてくるので，高さの値を入力します。あとはSPHEREやCYLINDERの場合と同様のデータ入力となります。

5. ROTATION (回転体)

〔TRANSFORMATION〕メニューで5を入力すると，回転体を記述することができます。

まず，

```
(1)
HEIGHT (E=EXIT) =
```

と聞いてくるので，中心点からの(1)の点の高さを入力してください。

次いで，

```
RADIUS =
```

と聞いてくるので，その点の半径を入力してください。そうすると，(1)が(2)に切り替わり，同様の

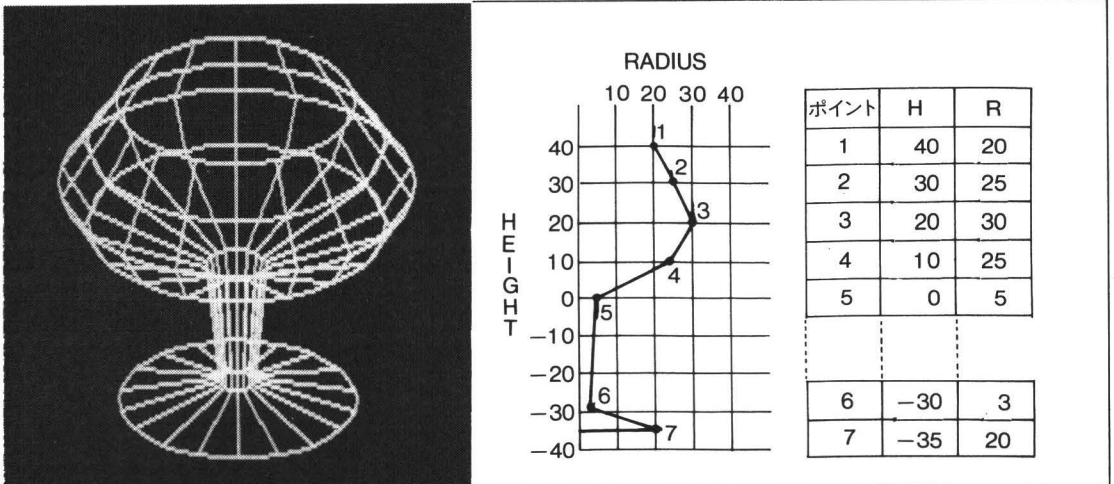



図3—14　回転体表現の例

質問が繰り返されるので，最大30個までの点を順次入力してください。

必要点の入力が終わり[E][CR]を入力すると，これまでのものと同様に，順次，COLOR 以降の入力データを求めてくるので，必要データを入力してください。

すべての記述が終われば，次の TRANS による図形データの入力を求めてきますが，終了の場合は[E][CR]を入力すると，〔PRODUCT〕のサブメニューに復帰します。

6．DELETE（消去）

〔TRANSFORMATION〕メニューで[6]を入力すると，入力した図形モデルを消去できます。

```
TRANS No.
```

と聞いてくるので，消去したい図形モデルの番号を入力してください。データが完全に消去されて，〔TRANSFORMATION〕のメニューに戻ります。

7．EXIT

〔TRANSFORMATION〕メニューの[7]で，〔PRODUCT〕のサブメニューに復帰します。

〔PRODUCT〕のサブメニューで[5]を入力すると，メインメニューに復帰します。

DISPLAY

メインメニューで DISPLAY 命令は，PRODUCT 命令によって，ワールド座標系に記述したワイヤーフレーム・モデルやサーフェイス・モデルを，スクリーン座標系でいろいろと変形してみようというものです。DISPLAY 命令に入ると，サブメニューとして，次の4つが表示されます。

```
1. NEW
2. CONDITION
3. WIRE  FRAME
4. EXIT
```

これらは，すべてワールド座標系に描かれた図形モデルを，何らかの形で見るためのものです。それぞれの命令について説明をしていきましょう。

NEW

[1]キーを入力して NEW 命令を選ぶと，次のように聞いてきます。

```
SCREEN  CLEAR  (Y/N)?
```

これは，他の命令によって描かれた画面上のスクリーン座標系に描かれたモデルを消すかどうか聞いているのです。消す場合は[Y]，そのままの場合は[N]キーを入力すると，DISPLAY メニューに復帰します。

CONDITION

[2]キーを入力して CONDITION 命令を選ぶと，まず，次のように聞いてきます。

```
〔SCREEN  CONDITION〕
PERSE  TRANS  (Y/N)  ?
```

透視変換する場合は[Y]，しない場合は[N]とキー入力します。透視変換する場合として，[Y]とキー入力すると，

```
Z  =
```

と，Z方向の消点位置を尋ねてきます。消点位置は，－の大きい値を入力すると，あまりパースペクティブはかかりません。逆に，－の小さい値を入力すると，かかりすぎます。物に応じて試して，適当な値を選び出してください。

Z値を入力すると，

```
AXIS  ROTATION  X  =
```

と，スクリーン座標系のX軸方向の回転角度を尋ねてくるので，X軸方向に正の回転角度を与えると，図3—15のように回転して表示されます。次いで，

```
AXIS  ROTATION  Y  =
```

と聞いてくるので同様の処理をしますと，角度分だけ回転して見えます。

物体を表示する位置は次の

```
SCREEN  MOVE  X  =
SCREEN  MOVE  Y  =
SCREEN  MOVE  Z  =
```

で指定します。スクリーンの動く方向は，図3—16のようになります。

これらの方向は，実際に物体を動かしてみて，画面上の位置関係を感覚的につかむことが大切だと思われます。目で確かめて早く自分のものにしてください。

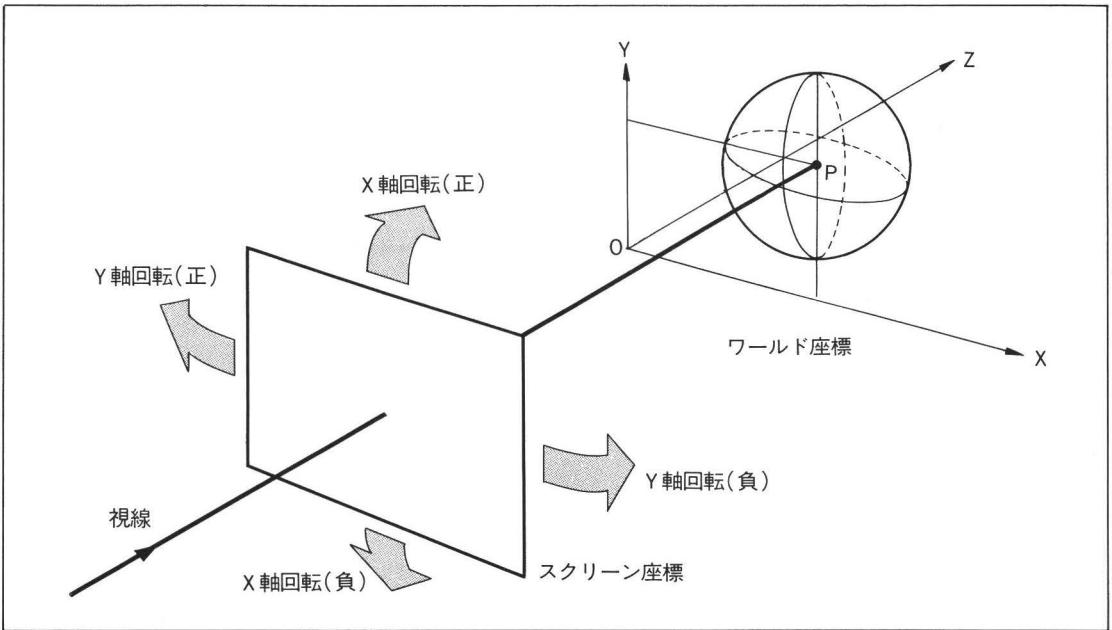



図3—15　スクリーン座標系のAXIS ROTATONの関係

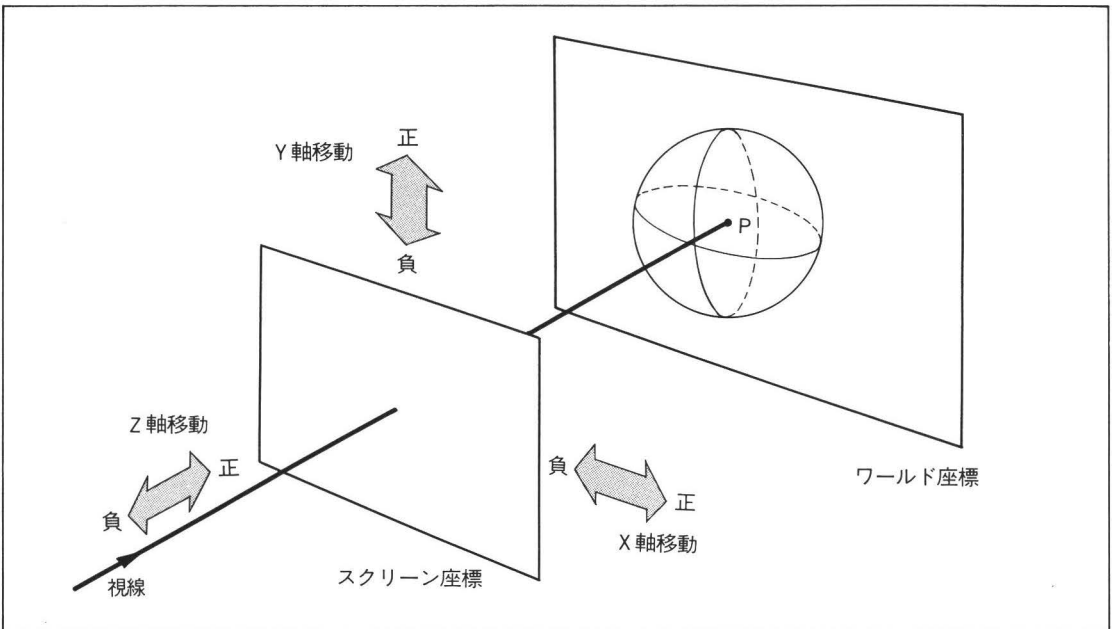



図3—16　スクリーン座標系のSCREEN　MOVEの関係

すべての入力が完了すると，〔DISPLAY〕メニューに復帰します。透視変換をしない場合は，スクリーン座標系の AXIS ROTATION 以降の命令を同様に入力してください。

WIRE FRAME

3キーを入力して WIRE FRAME 命令を選ぶと，まず，

```
〔WIRE  FRAME  DISPLAY〕
〔TRANS  No.
```

と聞いてくるので，表示する図形 No. を入力してください。[2]の CORDINATION 命令で指定されたスクリーンの位置で，入力図形が表示されます。

EXIT

[4]キーを入力するとメインメニューに戻ります。

LOAD

メインメニュー LOAD 命令は，(4)の SAVE 命令でファイルされた図形を呼び出すためのものです。

3次元デザインツール・プログラムのメニュー構造

メインメニュー　サブメニュー

- 1. PRODUCT
  - 1. POINT　面や物体のもとになっている点を入力する。
  - 2. PLANE　点と点をつなぎ合わせて線や面を作る。
  - 3. MANY P　PLANEで作り出した面を合わせて3次元物体を作る。
  - 4. TRANS FORMATION　ワールド座標系の中で10個まで登録できる。
    - 1. MANY P　3次元物体の回転，拡大，移動を行う
    - 2. SPHERE　球体の回転，拡大，移動を行う。
    - 3. CYLINDER　円柱の回転，拡大，移動を行う。
    - 4. CONE　円錐の回転，拡大，移動を行う。
    - 5. ROTATION　回転体の回転，拡大，移動を行う。
    - 6. DELETE　入力した図形モデルを消去する。
    - 7. EXIT　PRODUCTのサブメニューにもどる。
  - 5. EXIT　メインメニューにもどる。
- 2. DISPLAY
  - 1. NEW　画面をクリアする。
  - 2. CONDITION　透視図法で描くか，登録された図形の視点や倍率等の変化を与える。
  - 3. WIRE FRAME　SAVEされている図形をWIRE FRAMEで描き上げる。
  - 4. EXIT　メインメニューにもどる。
- 3. LOAD　外部記憶装置にファイルされている図形を呼び出す。
- 4. SAVE　図形を外部記憶装置にファイルする。
- NORMAL END

[3]キーを入力すると，

```
〔LOAD〕
〔LOAD  FILE  NAME  ('E' = EXIT)〕 =
```

と聞いてくるので，呼び出したいファイルネームを入力してください。呼び出しが完了するとメインメニューに復帰します。

SAVE

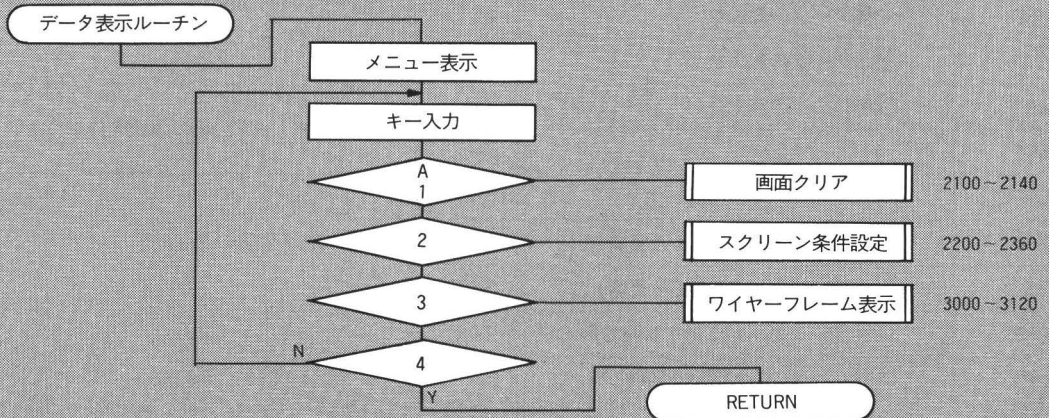
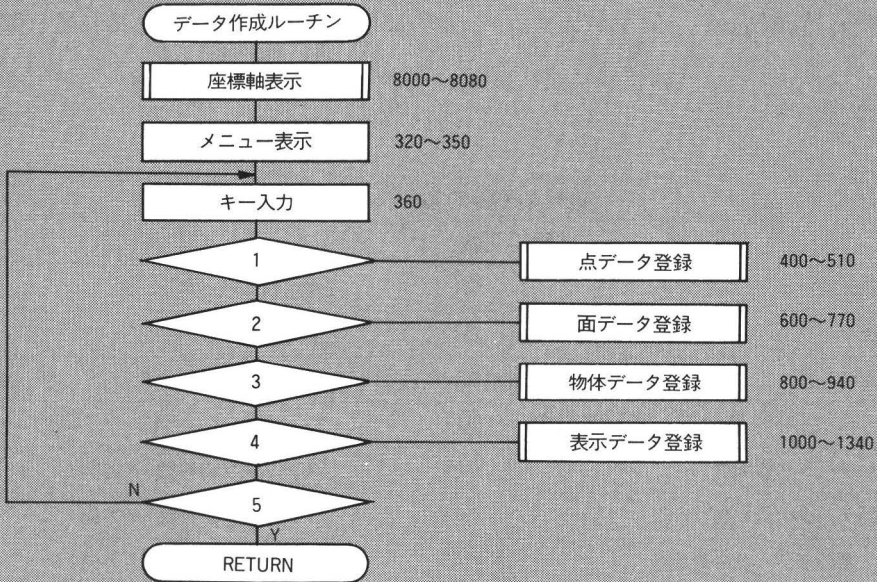
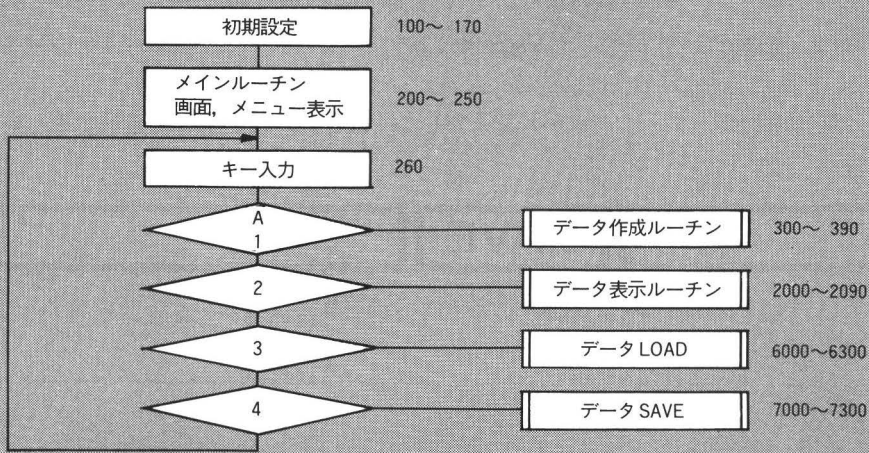
メインメニューで[4]キーを入力すると，作成した図形をSAVEしておくことができます。

画面上に，

```
〔SAVE〕
〔SAVE  FILE  NAME  ('E' = EXIT )〕 =
```

と表示されますので，必要なファイルディスクリプタを指定して13文字以内でファイル名を指定してください。

変数とその内容			
配列変数			
MD（ ）	物体データ	TR（ ）	表示物体データ
HP（ ）	面データ	RH（ ）	回転体の高さデータ
PT（ ）	点データ	RR（ ）	回転体の半径データ
変　数			
APT	点データの最大数	QAX	スクリーンの移動(Ｘ方向)
AHP	面データの最大数	QAY	スクリーンの移動(Ｙ方向)
BHP	１面の点データの最大数	QAZ	スクリーンの移動(Ｚ方向)
BBD	１物体の面データの最大数	MN	物体表示の変換の判定
ABD	物体データの最大数	SS	スクリーン回転の正弦成分
ATR	表示データの最大数	SF	
AR	回転体の点の最大数	TS	スクリーン回転の余弦成分
STP	回転角度のステップ数	TF	
STP1		SRX	物体の回転の正弦成分
XS	スクリーン中央の位置(Ｘ座標)	SRY	
YS	スクリーン中央の位置(Ｙ座標)	SRZ	
QRX	スクリーンのＹ軸に対する回転角	TRX	物体の回転の余弦成分
QRY	スクリーンのX－Z面での回転角	TRY	
PSZ	透視の判定と視点のＺ軸上の位置	TRZ	

3次元デザインツール・プログラムの流れ
初期設定
100〜 170
メインルーチン
画面，メニュー表示
200〜 250
キー入力
260
A
1
データ作成ルーチン
300〜 390
2
データ表示ルーチン
2000〜2090
3
データ LOAD
6000〜6300
4
データ SAVE
7000〜7300
データ作成ルーチン
座標軸表示
8000〜8080
メニュー表示
320〜350
キー入力
360
1
点データ登録
400〜510
2
面データ登録
600〜770
3
物体データ登録
800〜940
4
表示データ登録
1000〜1340
N
5
Y
RETURN
データ表示ルーチン
メニュー表示
キー入力
A
1
画面クリア
2100〜2140
2
スクリーン条件設定
2200〜2360
3
ワイヤーフレーム表示
3000〜3120
N
4
Y
RETURN

3次元デザインツール・プログラムリスト

```
1 '**************************************
2 '*                                    *
3 '*     3D -TOOL   ( WIRE FRAME )      *
4 '*                                    *
5 '*      Programmed By [ SAIMU ]       *
6 '*                                    *
7 '*           1985 . 4 . 1             *
8 '*                                    *
9 '**************************************
100 '----------  ｼｮｷ ｾｯﾃｲ  -------------------------
110 OPTION SCREEN 0:WIDTH 80,25,0,2:KLIST 0:KMODE 0
120 INIT:DEFINT A-N
130 APT=200:AHP=50:BHP=50:BBD=5:ABD=30:ATR=10:AR=30
140 DIM MD(BBD,ABD),HP(BHP,AHP),PT(2,APT)
150 DIM TR(ATR,13),RH(AR),RR(AR)
160 STP=20:STP1=10:XS=320:YS=100
170 QRX=-30:QRY=30:GOSUB 2340
200 '----------  ﾒｲﾝ ﾒﾆｭｰ  -------------------------
210 CLS:CONSOLE 0,25
220 PRINT" [ 3D-TOOL MAIN MENU ]":PRINT
230 PRINT"  1. PRODUCT":PRINT"  2. DISPLAY"
240 PRINT"  3. LOAD":PRINT"  4. SAVE"
250 PRINT"  5. NORMAL END"
260 LOCATE 2,24:PRINT" Push Key (1 - 5) ";
270 A=VAL(INKEY$(1))
280 ON A GOSUB 300,2000,6000,7000,295
290 GOTO 200
295 WIDTH 80,25:INIT:CLS 4:END
300 '----------  ﾌﾟﾛﾀﾞｸﾄ ﾙｰﾁﾝ  --------------------
310 GOSUB 8000:CONSOLE 0,25:CLS
320 PRINT" [ PRODUCT ]":PRINT:PRINT"  1. POINT"
330 PRINT"  2. PLANE":PRINT"  3. MANY P"
340 PRINT"  4. TRANSFORMATION":PRINT"  5. EXIT"
350 LOCATE 2,23:PRINT" Select Number";
360 A=VAL(INKEY$(1))
370 ON A GOSUB 400,600,800,1000,390
380 GOTO 310
390 RETURN 200
400 '----------  ﾎﾟｲﾝﾄ ﾌﾟﾛﾀﾞｸﾄ  -----------------
410 CLS:LOCATE 1,0:PRINT" [ POINT ]"
420 I=0:LOCATE 2,2:INPUT"<POINT No. ",N$
430 IF LEFT$(N$,1)="E" THEN RETURN
440 N=VAL(N$):IF N<=0 OR N>APT THEN 420
450 LOCATE 1,0:PRINT " [ POINT ]":PRINT
460 LOCATE 2,2:PRINT"<POINT No.";STR$(N);">";CHR$(5)
470 LOCATE 2,4:PRINT "X=";PT(0,N);CHR$(5)
480 LOCATE 2,5:PRINT "Y=";PT(1,N);CHR$(5)
```

```
490 LOCATE 2,6:PRINT "Z=";PT(2,N);CHR$(5)
500 LOCATE 4,4+I:INPUT "",PT(I,N)
510 I=I+1:IF I<3 THEN 450 ELSE 420
600 '---------- ﾌﾟﾚｲﾝ ﾌﾟﾛﾀﾞｸﾄ ------------------
610 CLS:LOCATE 1,0:PRINT" [ PLANE ]"
620 I=1:LOCATE 2,2:INPUT"<PLANE No. ",N$
630 IF LEFT$(N$,1)="E" THEN RETURN
640 N=VAL(N$):IF N<=0 OR N>AHP THEN 610
650 LOCATE 2,2:PRINT"<PLANE No.";STR$(N);">";CHR$(5)
660 LOCATE 2,3:PRINT"(";STR$(I);" )";CHR$(5)
670 LOCATE 2,4:PRINT"Point number (0=RETURN) =";
680 PRINT HP(I,N);CHR$(5)
690 LOCATE 28,4:INPUT"",N1
700 IF N1<=0 THEN 760 ELSE IF N1>APT THEN 650
710 HP(I,N)=N1:I2=N1:MN=0
720 GOSUB 5000
730 IF I=1 THEN LINE(X,Y)-(X,Y),PSET
740 IF I<>1 THEN LINE-(X,Y)
750 I=I+1:IF I<=BHP THEN 650
760 IF I<=2 THEN HP(0,N)=0:GOTO 620:ELSE HP(0,N)=I-1
770 I2=HP(1,N):GOSUB 5000:LINE -(X,Y):GOTO 620
800 '---------- ﾒﾆｰ ﾌﾟﾚｲﾝ ﾌﾟﾛﾀﾞｸﾄ --------------
810 CLS:LOCATE 1,0:PRINT" [ MANY P ]"
820 I=1:LOCATE 2,2:INPUT"<MANY P No. ",N$
830 IF LEFT$(N$,1)="E" THEN RETURN
840 N=VAL(N$):IF N<=0 OR N>BBD THEN 810
850 LOCATE 2,2:PRINT"<MANY P No.";
860 PRINT STR$(N);">";CHR$(5)
870 LOCATE 2,3:PRINT"(";STR$(I);" )"
880 LOCATE 2,4:PRINT"Plane number (0=RETURN) =";
890 PRINT MD(N,I);CHR$(5)
900 LOCATE 28,4:INPUT"",N1
910 IF N1<=0 THEN 950 ELSE IF N1>AHP THEN 850
920 MD(N,I)=N1:NN1=N1:MN=0:GOSUB 3250
930 I=I+1
940 IF I<=ABD THEN 850
950 MD(N,0)=I-1:GOTO 820
1000 '---------- ﾍﾝｶﾝ ｼﾞｮｳﾎｳ ------------------
1010 CLS:CONSOLE 0,25
1015 LOCATE 1,0:PRINT" [ TRANSFORMATION ]"
1020 LOCATE 2,2:INPUT"<TRANSFORMATION No. ",NT$
1025 IF LEFT$(NT$,1)="E" THEN CLS:RETURN
1030 NT=VAL(NT$):IF NT>ATR OR NT<=0 THEN 1000
1040 CLS:PRINT" [ TRANSFORMATION ]":PRINT
1050 PRINT"  1. MANY P  ":PRINT"  2. SPHERE"
1060 PRINT"  3. CYLINDER":PRINT"  4. CONE"
1070 PRINT"  5. ROTATION":PRINT"  6. DELETE"
```

```
1080 PRINT"  7. EXIT"
1090 LOCATE 2,24:PRINT" Select Number";
1100 A=VAL(INKEY$(1)):CLS
1110 ON A GOSUB 1400,1500,1600,1700,1800,1130,1140
1120 GOTO 1000
1130 TR(NT,0)=0
1140 RETURN 1000
1150 INPUT "  COLOR ( 1 - 7 ) = ",CL
1160 IF CL<0 OR CL>7 THEN 1150
1170 TR(NT,4)=CL:J=5
1180 A$="SCALE "
1190 A1$=" X = ":GOSUB 1320:GOSUB 1330
1200 A1$=" Y = ":GOSUB 1320:GOSUB 1330
1210 A1$=" Z = ":GOSUB 1320:GOSUB 1330
1220 A$="ROTATE "
1230 A1$=" X = ":GOSUB 1320:GOSUB 1340
1240 A1$=" Y = ":GOSUB 1320:GOSUB 1340
1250 A1$=" Z = ":GOSUB 1320:GOSUB 1340
1260 A$="MOVE "
1270 A1$=" X = ":GOSUB 1320:GOSUB 1340
1280 A1$=" Y = ":GOSUB 1320:GOSUB 1340
1290 A1$=" Z = ":GOSUB 1320:GOSUB 1340
1300 CLS:MN=1:SWAP NT,N:GOSUB 3060
1310 SWAP NT,N:COLOR7:RETURN
1320 PRINT"  TRANS ";A$;A1$;:INPUT" ",X:RETURN
1330 IF X<=0 THEN X=1
1340 TR(NT,J)=X:J=J+1:RETURN
1400 '----------- ﾄﾗﾝｽ ﾒﾆｰ ﾌﾟﾚｲﾝ -----------------
1410 CLS:LOCATE 1,0:PRINT" [ TRANS MANY P ]"
1420 LOCATE 2,2:INPUT" MANY P NUMBER = ",N
1430 IF N<=0 THEN RETURN ELSE IF N>BBD THEN 1430
1440 TR(NT,0)=1:TR(NT,1)=N:GOTO 1150
1500 '----------- ｷｭｳ ﾉ ﾍﾝｶﾝ --------------------
1510 CLS:LOCATE 1,0:PRINT" [ TRANS SPHERE ]"
1520 LOCATE 2,2:INPUT"RADIUS = ",R
1530 IF R<=0 THEN 1510
1540 TR(NT,0)=2:TR(NT,1)=R:GOTO 1150
1600 '----------- ｴﾝﾁｭｳ ﾉ ﾍﾝｶﾝ ------------------
1610 CLS:LOCATE 1,0:PRINT" [ TRANS CYLINDER ]"
1620 LOCATE 2,2:INPUT"RADIUS = ",R
1630 IF R<=0 THEN 1620
1640 INPUT"  HEIGHT = ",RH:IF RH<=0 THEN 1640
1650 TR(NT,0)=3:TR(NT,1)=R:TR(NT,2)=RH:GOTO 1150
1700 '----------- ｴﾝｽｲ ﾉ ﾍﾝｶﾝ -------------------
1710 CLS:LOCATE 1,0:PRINT" [ TRANS CONE ]"
1720 LOCATE 2,2:INPUT"UPPER RADIUS = ",R1
1730 IF R1<0 THEN 1720
```

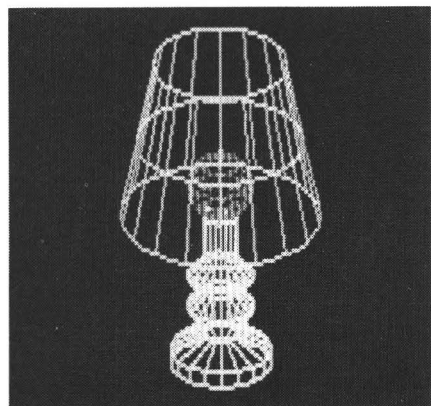
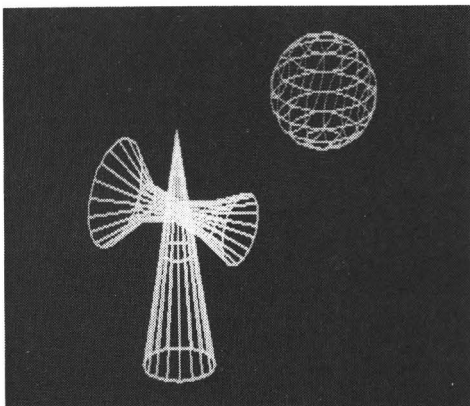
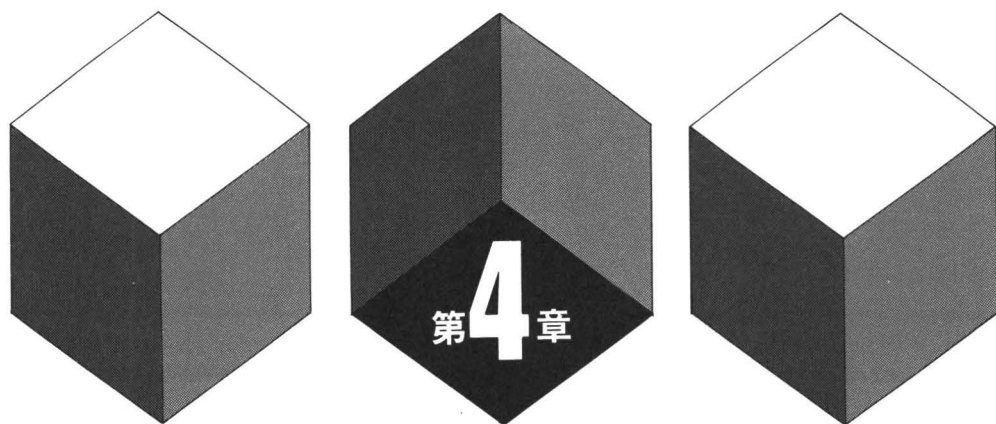
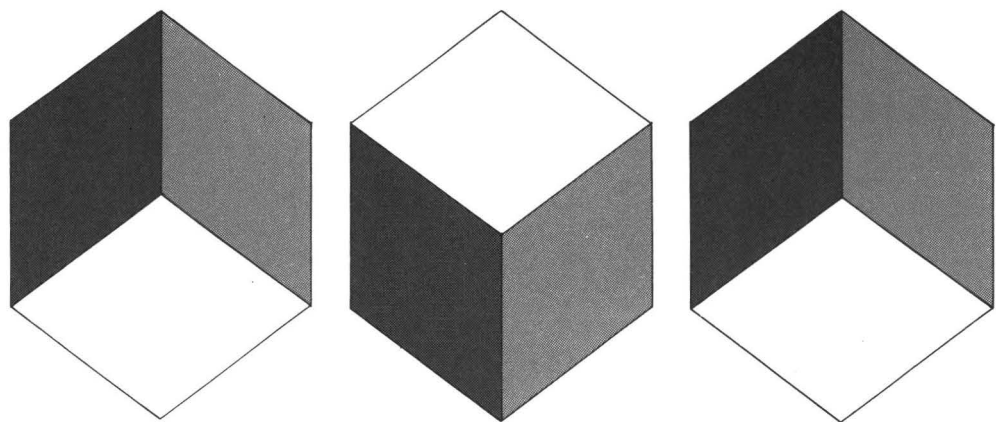
```
1740 INPUT"  LOWER RADIUS = ",R2
1750 IF R2<0 THEN 1740 ELSE IF R1=R2 THEN 1640
1760 INPUT"  HEIGHT = ",RH:IF RH<=0 THEN 1760
1770 TR(NT,0)=4:TR(NT,1)=R1:TR(NT,2)=R2
1780 TR(NT,3)=RH:GOTO 1150
1800 '----------  ｶｲﾃﾝﾀｲ ﾉ ﾍﾝｶﾝ  ---------------
1810 CLS:LOCATE 1,0:PRINT" [ TRANS ROTATION ]"
1820 I=1:RR(0)=0:RH(0)=0
1830 LOCATE 2,2:PRINT"(";STR$(I);" ) "
1840 LOCATE 2,3:PRINT"HEIGHT (E=EXIT) = ";
1850 INPUT"",RH$
1860 IF RH$="E" THEN 1930 ELSE RH=VAL(RH$)
1870 LOCATE 2,4:PRINT"RADIUS = ";
1880 INPUT"",RR
1890 IF RR<0 THEN 1870
1900 RR(I)=RR:RH(I)=RH
1910 I=I+1
1920 IF I=<AR THEN 1830
1930 IF I=1THEN RETURN ELSE TR(NT,0)=5:TR(NT,1)=I-1
1940 GOTO 1150
2000 '----------  ﾃﾞｨｽﾌﾟﾚｲ ﾙｰﾁﾝ  ---------------
2010 CLS:CONSOLE 0,25
2020 PRINT" [ DISPLAY ]":PRINT
2030 PRINT"  1. NEW":PRINT"  2. CONDITION"
2040 PRINT"  3. WIRE FRAME":PRINT"  4. EXIT"
2050 LOCATE 2,24:PRINT" Select Number";
2060 A=VAL(INKEY$(1)):CLS
2070 ON A GOSUB 2100,2200,3000,2090
2080 COLOR 7:GOTO 2010
2090 RETURN 200
2100 '----------  ｽｸﾘｰﾝ ｸﾘｱ  ------------------
2110 CLS:LOCATE 1,1:PRINT" [ SCREEN CLEAR ]"
2120 PRINT"  SCREEN CLEAR (Y/N) ? ";
2130 A$=INKEY$(1):IF A$="Y" THEN GOSUB 8000:RETURN
2140 IF A$="N" THEN RETURN ELSE 2120
2200 '----------  ｽｸﾘｰﾝ ｺﾝﾃﾞｨｼｮﾝ  ---------------
2210 CLS:LOCATE 1,0:PRINT" [ SCREEN CONDITION ]"
2220 PRINT"  PERSE TRANS (Y/N) ? ";
2230 A$=INKEY$(1):CLS
2240 IF A$="N" THEN PSZ=0:GOTO 2280
2250 IF A$<>"Y" THEN 2220
2260 LOCATE 1,0:PRINT" [ SCREEN CONDITION ]"
2270 INPUT"  Z = ",PSZ:IF PSZ=0 THEN 2260
2280 LOCATE 1,0:PRINT" [ SCREEN CONDITION ]"
2290 LOCATE 2,3:INPUT"AXIS ROTATION X = ",QRX
2300 INPUT"  AXIS ROTATION Y = ",QRY
2310 INPUT"  SCREEN MOVE X = ",QAX
```

```
2320 INPUT"  SCREEN MOVE Y = ",QAY
2330 INPUT"  SCREEN MOVE Z = ",QAZ
2340 SS=SIN(RAD(QRX)):SF=SIN(RAD(QRY))
2350 TS=COS(RAD(QRX)):TF=COS(RAD(QRY))
2360 GOSUB 8000:RETURN
3000 '----------  ﾜｲﾔｰ ﾌﾚｰﾑ ﾃﾞｨｽﾌﾟﾚｲ  ------------
3010 MN=1
3020 CLS:LOCATE 1,0:PRINT" [ WIRE FRAME DISPLAY ]"
3030 LOCATE 2,2:INPUT"<TRANS No. ",N$
3040 IF N$="E" THEN RETURN
3050 N=VAL(N$):IF N<=0 OR N>ATR THEN 3020
3060 COLOR TR(N,4)
3070 ON TR(N,0) GOTO 3200,3400,3600,3800,4000
3080 COLOR 7
3090 A$="<TRANS No":B$="> is miss input "
3100 C$=" !! Push any key !!"
3110 LOCATE 2,2:PRINT A$;N;B$;C$;
3120 A$=INKEY$(1):RETURN
3200 '----------  ﾜｲﾔｰ ﾌﾚｰﾑ ﾃﾞｨｽﾌﾟﾚｲ 1  ----------
3210 N0=TR(N,1):N1=MD(N0,0)
3220 FOR I1=1 TO N1
3230   NN1=MD(N0,I1):GOSUB 3250
3240 NEXT:RETURN
3250 I2=HP(1,NN1):GOSUB 5000:LINE (X,Y)-(X,Y),PSET
3260 FOR J=2 TO HP(0,NN1)
3270   I2=HP(J,NN1):GOSUB 5000:LINE -(X,Y)
3280 NEXT
3290 I2=HP(1,NN1):GOSUB 5000:LINE -(X,Y):RETURN
3400 '----------  ﾜｲﾔｰ ﾌﾚｰﾑ ﾃﾞｨｽﾌﾟﾚｲ 2  ----------
3410 R=TR(N,1)
3420 FOR J1=0 TO 180 STEP STP
3430   X0=R*COS(RAD(J1)):R1=R*SIN(RAD(J1))
3440   X=X0:Y=R1:Z=0
3450   GOSUB 5020:LINE(X,Y)-(X,Y),PSET
3460   FOR J2=0 TO 360 STEP STP1
3470     X=X0:Y=R1*COS(RAD(J2)):Z=R1*SIN(RAD(J2))
3480     GOSUB 5020:LINE -(X,Y)
3490   NEXT
3500 NEXT
3510 FOR J1=0 TO 180 STEP STP
3520   Y0=R*COS(RAD(J1)):R1=R*SIN(RAD(J1))
3530   X=R1:Y=Y0:Z=0
3540   GOSUB 5020:LINE(X,Y)-(X,Y),PSET
3550   FOR J2=0 TO 360 STEP STP1
3560     X=R1*COS(RAD(J2)):Y=Y0:Z=R1*SIN(RAD(J2))
3570     GOSUB 5020:LINE -(X,Y)
3580   NEXT
```

```
3590 NEXT:RETURN
3600 '----------  ﾜｲﾔｰ ﾌﾚｰﾑ ﾃﾞｨｽﾌﾟﾚｲ 3  ----------
3610 R=TR(N,1):RH=TR(N,2)
3620 FOR ZJ1=0 TO RH STEP RH/2
3630   X=R:Y=ZJ1:Z=0
3640   GOSUB 5020:LINE(X,Y)-(X,Y),PSET
3650   FOR J2=0 TO 360 STEP STP1
3660     X=R*COS(RAD(J2)):Y=ZJ1:Z=R*SIN(RAD(J2))
3670     GOSUB 5020:LINE -(X,Y)
3680   NEXT
3690 NEXT
3700 FOR J=0 TO 360 STEP STP
3710   X=R*COS(RAD(J)):Y=RH:Z=R*SIN(RAD(J))
3720   GOSUB 5020:X1=X:Y1=Y
3730   X=R*COS(RAD(J)):Y=0:Z=R*SIN(RAD(J))
3740   GOSUB 5020:X2=X:Y2=Y
3750   LINE (X1,Y1)-(X2,Y2),PSET
3760 NEXT:RETURN
3800 '----------  ﾜｲﾔｰ ﾌﾚｰﾑ ﾃﾞｨｽﾌﾟﾚｲ 4  ----------
3810 R1=TR(N,1):R2=TR(N,2):RH=TR(N,3)
3820 FOR ZJ1=0 TO RH STEP RH/2
3830   R=R2+(R1-R2)*(RH-ZJ1)/RH
3840   X=R:Y=ZJ1:Z=0
3850   GOSUB 5020:LINE(X,Y)-(X,Y),PSET
3860   FOR J2=0 TO 360 STEP STP1
3870     X=R*COS(RAD(J2)):Y=ZJ1:Z=R*SIN(RAD(J2))
3880     GOSUB 5020:LINE -(X,Y)
3890   NEXT
3900 NEXT
3910 FOR J=0 TO 360 STEP STP
3920   X=R1*COS(RAD(J)):Y=0:Z=R1*SIN(RAD(J))
3930   GOSUB 5020:X1=X:Y1=Y
3940   X=R2*COS(RAD(J)):Y=RH:Z=R2*SIN(RAD(J))
3950   GOSUB 5020:X2=X:Y2=Y
3960   LINE (X1,Y1)-(X2,Y2),PSET
3970 NEXT:RETURN
4000 '----------  ﾜｲﾔｰ ﾌﾚｰﾑ ﾃﾞｨｽﾌﾟﾚｲ 5  ----------
4010 FOR J1=1 TO TR(N,1):R=RR(J1)
4020   Y0=RH(J1):Y=Y0:X=R:Z=0
4030   GOSUB 5020:LINE(X,Y)-(X,Y),PSET
4040   FOR J2=0 TO 360 STEP STP1
4050     X=R*COS(RAD(J2)):Y=Y0:Z=R*SIN(RAD(J2))
4060     GOSUB 5020:LINE -(X,Y)
4070   NEXT
4080 NEXT
4090 FOR J1=0 TO 360 STEP STP
4100   X=RR(1)*COS(RAD(J1)):Y=RH(1)
```

```
4110    Z=RR(1)*SIN(RAD(J1))
4120    GOSUB 5020:LINE(X,Y)-(X,Y),PSET
4130    FOR J2=2 TO TR(N,1)
4140      X=RR(J2)*COS(RAD(J1)):Y=RH(J2)
4150      Z=RR(J2)*SIN(RAD(J1))
4160      GOSUB 5020:LINE -(X,Y)
4170    NEXT
4180 NEXT:RETURN
5000 '---------- トランス ルーチン ---------------------
5010 X=PT(0,I2):Y=PT(1,I2):Z=PT(2,I2)
5020 IF MN=1 THEN GOSUB 5500
5030 X=X-QAX:Y=Y-QAY:Z=Z-QAZ
5040 XX=X*TF+Z*SF
5050 YY=X*SS*SF+Y*TS-Z*SS*TF
5060 ZZ=-X*TS*SF+Y*SS+Z*TS*TF
5070 X=XX:Y=YY:Z=ZZ
5080 IF PSZ=0 THEN 5100
5090 PS=1-Z/PSZ:X=X/PS:Y=Y/PS
5100 X=2*X+XS:Y=YS-Y:RETURN
5500 '---------- トランス ルーチン ---------------------
5510 K=5:GOSUB 5630:GOSUB 5620:GOSUB 5630
5520 GOSUB 5540:GOSUB 5630
5530 X=X+TX:Y=Y+TY:Z=Z+TZ:RETURN
5540 XA=X*TRY*TRZ+Y*(SRX*SRY*TRZ+TRX*SRZ)
5550 XB=Z*(TRX*SRY*TRZ-SRX*SRZ)
5560 XX=XA-XB
5570 YA=-X*TRY*SRZ-Y*(SRX*SRY*SRZ-TRX*TRZ)
5580 YB=Z*(TRX*SRY*SRZ+SRX*TRZ)
5590 YY=YA+YB
5600 ZZ=X*SRY-Y*SRX*TRY+Z*TRX*TRY
5610 X=XX:Y=YY:Z=ZZ:RETURN
5620 X=X*TX:Y=Y*TY:Z=Z*TZ:RETURN
5630 TX=TR(N,K):TY=TR(N,K+1):TZ=TR(N,K+2)
5640 IF K<>8 THEN 5690
5650 SRX=SIN(RAD(TX)):SRY=SIN(RAD(TY))
5660 SRZ=SIN(RAD(TZ))
5670 TRX=COS(RAD(TX)):TRY=COS(RAD(TY))
5680 TRZ=COS(RAD(TZ))
5690 K=K+3:RETURN
6000 '---------- ロード -------------------------
6010 CLS:LOCATE 1,0:PRINT" [ LOAD ]":LOCATE 2,2
6020 INPUT"[LOAD FILE NAME ('E'= EXIT )] = ",T$
6030 IF T$="E" THEN RETURN
6040 OPEN "I",1,T$
6050 FOR I=0 TO ATR
6060   FOR J=0 TO 13
6070     INPUT #1,TR(I,J)
```

```
6080   NEXT
6090 NEXT
6100 FOR I=0 TO BBD
6110   INPUT #1,MD(I,0)
6120   FOR J=1 TO MD(I,0)
6130     INPUT #1,MD(I,J)
6140   NEXT
6150 NEXT
6160 FOR I=0 TO AHP
6170   INPUT #1,HP(0,I)
6180   FOR J=1 TO HP(0,I)
6190     INPUT #1,HP(J,I)
6200   NEXT
6210 NEXT
6220 FOR I=0 TO APT
6230   FOR J=0 TO 2
6240     INPUT #1,PT(J,I)
6250   NEXT
6260 NEXT
6270 FOR I=0 TO AR
6280   INPUT #1,RR(I),RH(I)
6290 NEXT
6300 CLOSE #1:RETURN
7000 '----------  ｾｰﾌﾞ  -------------------------
7010 CLS:LOCATE 1,0:PRINT" [ SAVE ]":LOCATE 2,2
7020 INPUT"[SAVE FILE NAME ('E'= EXIT )] = ",T$
7030 IF T$="E" THEN RETURN
7040 OPEN "O",1,T$
7050 FOR I=0 TO ATR
7060   FOR J=0 TO 13
7070     PRINT #1,TR(I,J)
7080   NEXT
7090 NEXT
7100 FOR I=0 TO BBD
7110   PRINT #1,MD(I,0)
7120   FOR J=1 TO MD(I,0)
7130     PRINT #1,MD(I,J)
7140   NEXT
7150 NEXT
7160 FOR I=0 TO AHP
7170   PRINT #1,HP(0,I)
7180   FOR J=1 TO HP(0,I)
7190     PRINT #1,HP(J,I)
7200   NEXT
7210 NEXT
7220 FOR I=0 TO APT
7230   FOR J=0 TO 2
```

```
7240      PRINT #1,PT(J,I)
7250    NEXT
7260 NEXT
7270 FOR I=0 TO AR
7280    PRINT #1,RR(I),RH(I)
7290 NEXT
7300 CLOSE #1:RETURN
8000 '---------- ｻﾞﾋｮｳｼﾞｸ ｦ ｴｶﾞｸ ---------------
8010 CLS 4:RESTORE 9000
8020 READ CL,X,Y,Z:GOSUB 8080
8030 LINE(X,Y)-(X,Y),PSET
8040 READ X,Y,Z:IF X=-99 THEN COLOR 7:RETURN
8050 IF X=99 THEN 8020
8060 GOSUB 8080
8070 LINE -(X,Y):GOTO 8040
8080 GOSUB 5030:COLOR CL:X=X+260:Y=Y+70:RETURN
9000 '---------- ｻﾞﾋｮｳｼﾞｸ ﾉ ﾃﾞｰﾀ ---------------
9010 DATA 1,-8,0,0,-6,0,0,99,99,99
9020 DATA 1,-4,0,0,-2,0,0,99,99,99,1,0,0,0,20,0,0
9030 DATA 17,2,0,20,0,0,17,-2,0,99,99,99
9040 DATA 1,22,2,0,26,-2,0,99,99,99
9050 DATA 1,22,-2,0,26,2,0,99,99,99
9060 DATA 2,0,-8,0,0,-6,0,99,99,99
9070 DATA 2,0,-4,0,0,-2,0,99,99,99,2,0,0,0,0,20,0
9080 DATA -2,17,0,0,20,0,2,17,0,99,99,99
9090 DATA 2,-2,26,0,0,25,0,2,26,0,0,25,0
9100 DATA 0,22,0,99,99,99
9110 DATA 4,0,0,-8,0,0,-6,99,99,99
9120 DATA 4,0,0,-4,0,0,-2,99,99,99,4,0,0,0,0,0,20
9130 DATA 0,2,17,0,0,20,0,-2,17,99,99,99
9140 DATA 4,-2,2,24,2,2,24,-2,-2,24
9150 DATA 2,-2,24,-99,-99,-99
```

マッピング処理技術

図形記述のためのプリミティブの設定

マッピング処理を行うための図形記述のプリミティブ（基本形状）は，次の5タイプを用意してあります。

(1) 球体（sphere：本書では SPHERE）

(2) 円柱（circular cylinder：CYLINDER と略）

(3) 円錐（circular cone：CONE と略）

(4) 直方体（rectangular parallelopiped：RECTANGULAR と略）

(5) 回転体（body of rotation：ROTATION と略）

それぞれの記述は，第3章の3次元デザインツール・プログラムに登録している図形記述のためのものと同様に，ワイヤーフレーム・モデルで記述しています。

球体（SPHERE）

SPHERE は球体を表します。形状記述指定は半径 R で指定します。

基本的にはワイヤーフレーム・モデルの記述ですので，X軸上 $-R$ から R までのX軸に対して垂直な面での切り口（円）と，Y軸上 $-R$ から R までの1点でY軸に対して垂直な面の切り口（円）で表します。

X軸に対して垂直な切り口の円の方程式は，

$$\begin{cases} y=\sqrt{R^2-x_1{}^2}\cos t \\ z=\sqrt{R^2-x_1{}^2}\sin t \end{cases}$$

Y軸に対して垂直な切り口の円の方程式は，

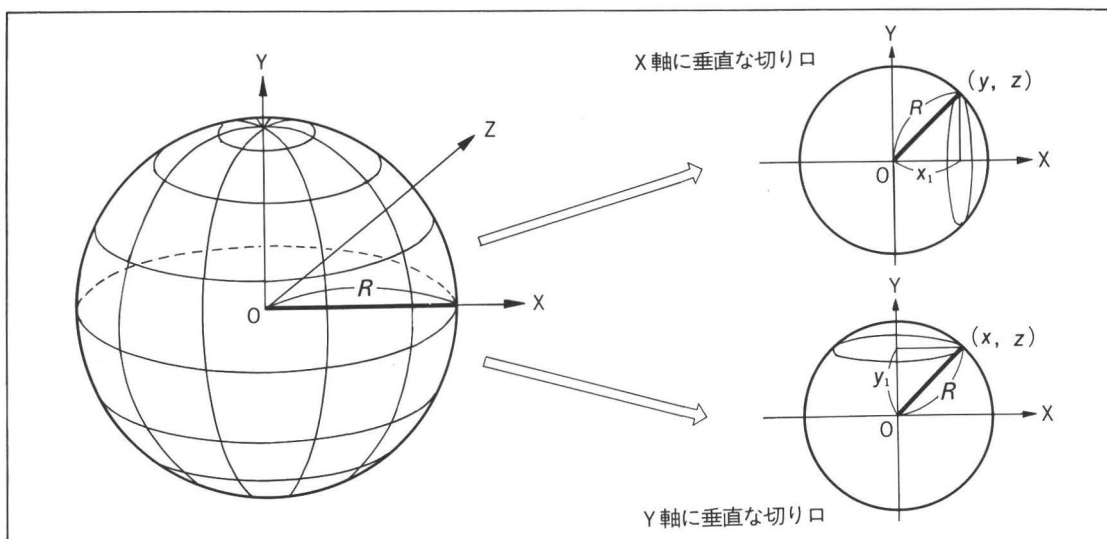



図4－1　球体表現

$$\begin{cases} x=\sqrt{R^2-y_1{}^2}\cos t \\ z=\sqrt{R^2-y_1{}^2}\sin t \end{cases}$$

と表すことができます。

これらの円の切り口の位置を変えて連続させることにより，球面を表現することができます。

円柱(CYLINDER)

CYLINDERは円柱を表します。形状記述指定は，円柱の半径 R と高さのY座標 H で表します。

これも基本的には，ワイヤーフレーム・モデルによる記述ですので，$y=0, H$ のところで半径 R の円を表現した後，円柱の側面に直線を引かせています（つまり，上下の円周を分割した点を結ばせています）。

円柱の方程式は，次のように表せます。

$$\begin{cases} y=0 \text{（または } y=H\text{）} \\ x=R\cos t \\ y=R\sin t \end{cases}$$

円錐(CONE)

CONEは円錐を表します。形状記述指定は $y=0$ での半径 R_1 と，円錐の高さ H で表します。これもワイヤーフレーム・モデルなので，$y=0$ および H で円を描かした後，円錐の側面に直線を引かせています。

円錐の方程式は，次のように表せます。

$$\begin{cases} y=0 \\ x=R_1\cos t \\ z=R_1\sin t \end{cases} \qquad \begin{cases} y=H \\ x=R_2\cos t \\ z=R_2\sin t \end{cases}$$

（ただし　$R_2=0$ の場合は円錐，$R_2\neq 0$ の場合は円錐台）

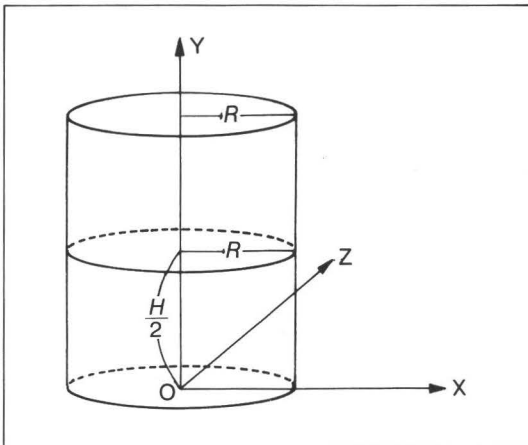



図4－2　円柱表現

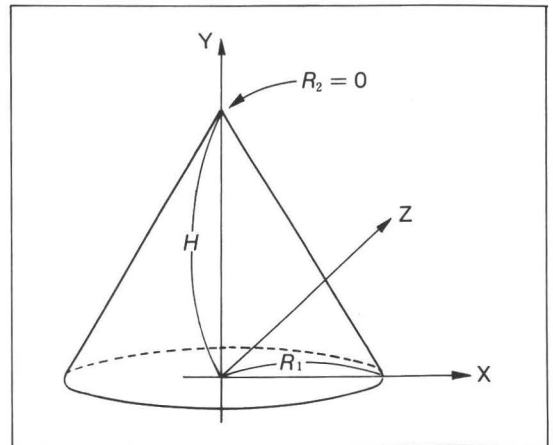



図4－3　円錐表現

直方体(RECTANGULAR)

RECTANGULAR は直方体を表します。形状記述指定は，直方体の縦方向の長さ H，横幅 W，奥行 D で表現します。直方体を構成する8頂点は，図4—4のように表現できます。

それぞれの頂点を結ぶと，直方体を表すことができます。本書のマッピング処理でのアルゴリズムでは，正方形全体を表現せずに，正面，立面，側面の3面のみを表現しています。

回転体(ROTATION)

ROTATION は回転体を表します。

基本的には，円錐を連続的に表現した形式をとっています。したがって，まず基点の $y=0$ のとき，半径を $RX(0)$ として入力すると，次のように書き表せます。

$$\begin{cases} y=0 \\ x=RX(0)\cos t \\ z=RX(0)\sin t \end{cases}$$

Y軸に移動量 $RY(1)$，そのときの半径を $RX(1)$ とすると，

$$\begin{cases} y=RY(1) \\ x=RX(1)\cos t \\ z=RX(1)\sin t \end{cases}$$

と書き表せます。順次，Y軸上の移動量を $RY(n)$，そのときの半径を $RX(n)$ とすると，

$$\begin{cases} y=RY(n) \\ x=RX(n)\cos t \\ z=RX(n)\sin t \end{cases}$$

と表現できます。

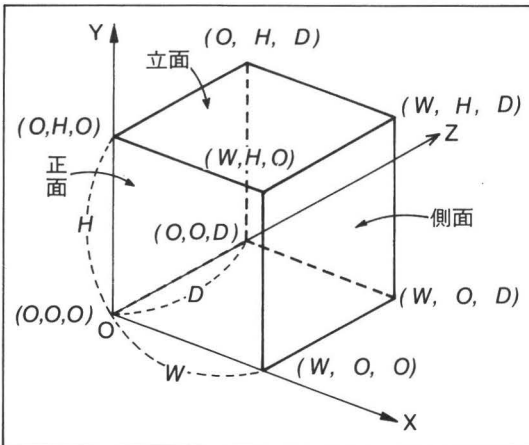



図4—4　直方体表現

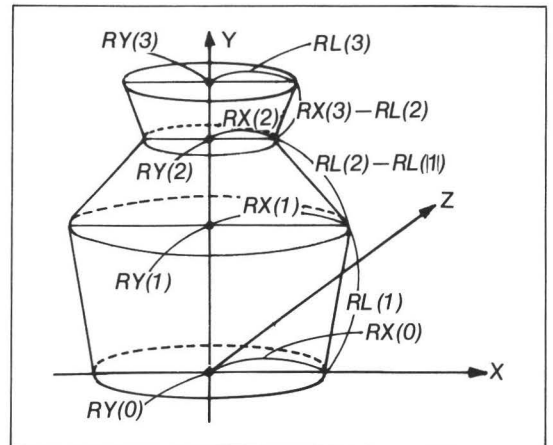



図4—5　回転体表現

以上のプリミティブをそれぞれ指定した後，それぞれのテクスチャの画面を，プリミティブの表面にはりつけていきます。

コンピュータ画面上のスクリーンの指定

X1ターボは，低解像画面（640×200ドット）指定の場合，2画面持っています。そこで，マッピングを行うための表面パターンは，LOAD命令でいったんスクリーン2に呼び出してから行います。スクリーン0のほうには，マッピングを行うためのプリミティブの形状を記述しています。

X1シリーズには，640×200ドットの画面が1画面しかありませんので，320×200ドットの画面を使います。そのために，マッピング・プログラムを少し変更して使用してください。変更場所は，各プログラムの最後に記載してあります。

なお，スクリーン1に記述するためのパターンのデザインは，第2章のデザインツール・プログラムで作成したものを使用するわけですが，メインメニューから 1. SCREEN　MENU を選択した後，X1ターボでご使用の読者は，1の画面で描いたうえで画面セーブを，X1シリーズご使用の読者は，2の画面で描いたものを必ず使用してください。

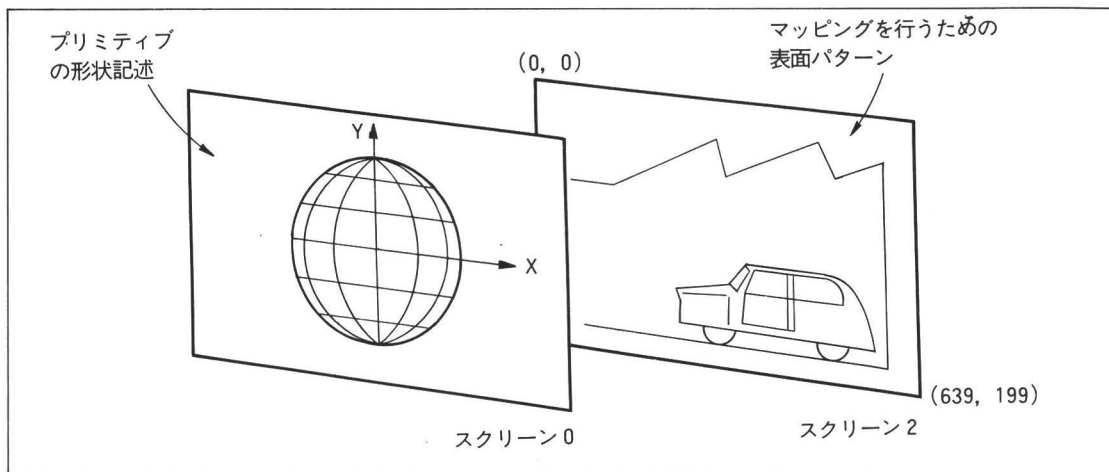



図4—6　X1ターボのマッピング画面

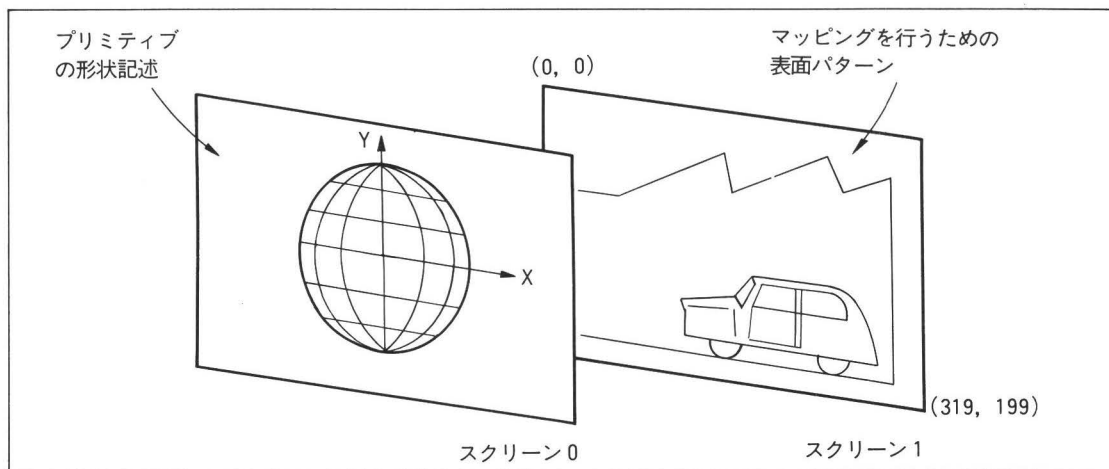



図4—7　X1シリーズのマッピング画面

マッピング処理のための透視変換と座標変換

マッピング処理を施すための座標記述は，平行投影で描いてあります。そのため，パースペクティブはかかりませんのでご注意ください。

平行投影は，一般的に製図等でよく用いられているものです。物体そのものが投影面に対し平行に投影されるため，投影面上でのX，Yの座標値はまったく変わりません。

物体形状の記述は，第3章の「3次元デザインツール」と同様に，左手系で描いてあります。スクリーン座標は平行透視のためZ軸は関係しないので固定していると考えて，ワールド座標軸に描かれた物体そのものの座標が表されることになります。そのため，スクリーン上にも物体指定そのものの座標形状が記述されます。

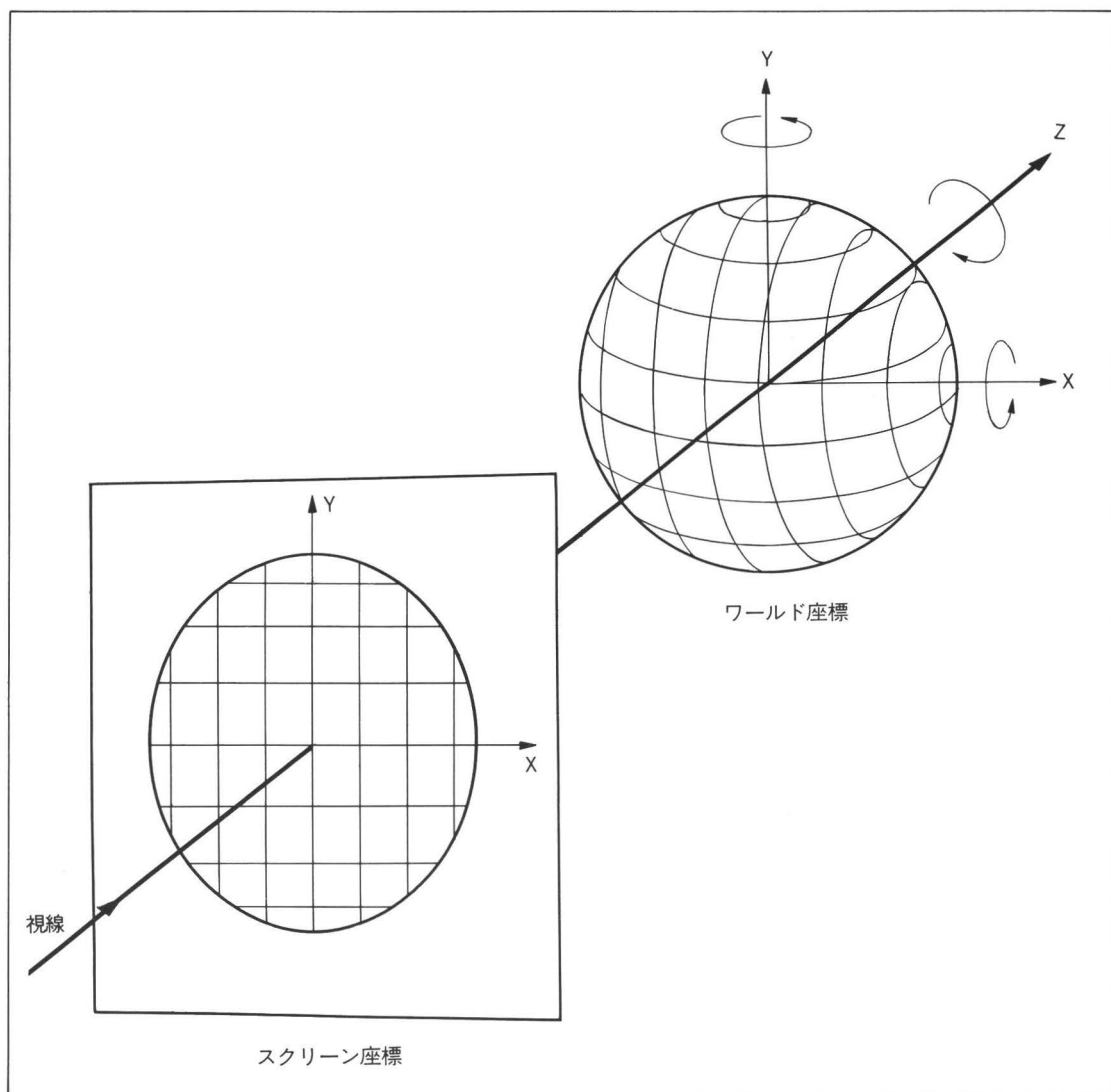



図4－8　ワールド座標／スクリーン座標と視線の関係

球体へのマッピング・プログラム

球体へのマッピング技術は，テクスチャパターン画面に描かれた画面上のピクセル情報を，球面のそれぞれの座標値へ変換させて指示していきます。

テクスチャパターン画面から球面への変換式は，テクスチャパターンのY方向の倍率 SC_Y を求めます。テクスチャ画面のY方向の長さを L_Y とすると，マッピングのための球の円周の半分は πR であるので，

$$SC_Y = \pi R / L_Y$$

と書き表せます。

次いで，テクスチャパターン画面のY方向の長さと対応する球体の点 P_Y を求めます。

$$R_Y = R \cos \theta_1$$

$$\theta_1 = J / L_Y \cdot \pi \text{（ラジアン）}$$

次に PY 軸に対して垂直な面の切り口の円の半径 LL を求めます。

$$LL = \sqrt{R^2 - PY^2}$$

今度は，マッピングする側の画面（スクリーン0側）のX方向の倍率 SC_X を求めると，

$$SC_X = \pi \cdot LL / L_X$$

と書き表せます。

次に，テクスチャパターン画面のX方向の点と対応する球の点 P_X, P_Z を求めると，

$$P_X = LL \cos \theta_2$$

$$P_Z = LL \sin \theta_2$$

ただし $\theta_2 = I / L_X \cdot \pi$（ラジアン）

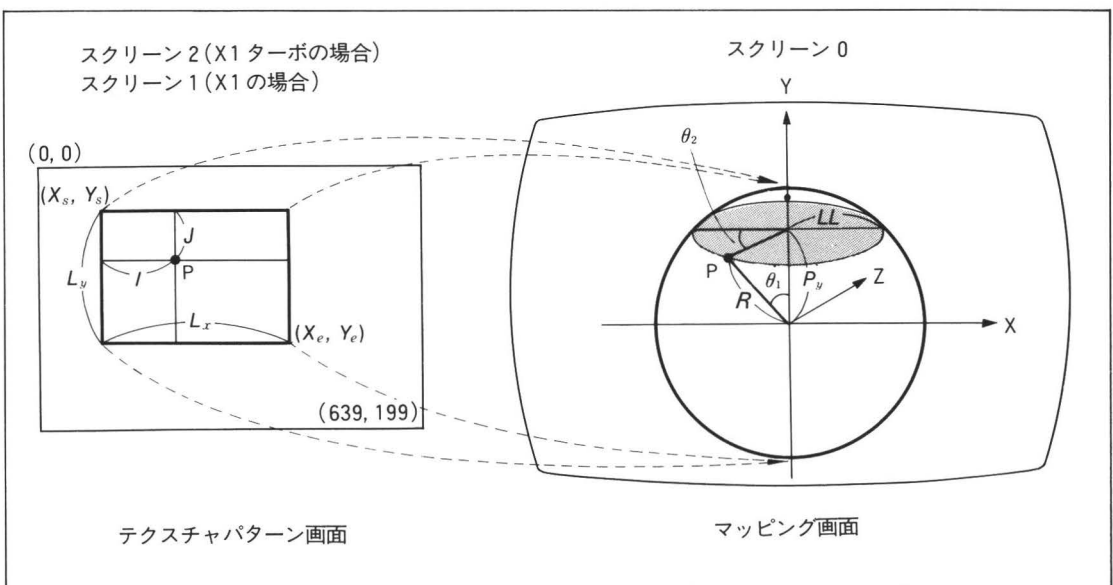



図4―9　球体へのマッピング

と書き表すことができます。

以上の計算式から，テクスチャパターン上の点 (I, J) の色を決め，P_X, P_Y, P_Z を座標変換したスクリーン上にセットすると，球面へのマッピングが行われます。

プログラムの操作方法

このプログラムを入力して実行（RUN）させると，画面上に，

```
〔SCREEN  LOAD〕
FILE  NAME  ?
```

と聞いてくるので，第2章のデザインツール・プログラムで作成した画像データのうち，マッピングしたい画像のファイル名を入力して，画面上に読み込んでください（スクリーン1に読み込まれます）。読み込みが完了すると，いよいよマッピング処理のための物体形状の記述に入ります。画面上には，

```
キュウ  ノ  ハンケイ  =
```

と聞いてきます。このマッピング・プログラムは，球体への処理技術なので，半径の数値を入力してください。

次いで，球を画面でのこの位置に描くかの設定を行います。コマンドは，

```
X――MOVE   =
Y――MOVE   =
```

と聞いてくるので，ワールド座標軸に描く球体を，原点からどこに描くのか，中心のX，Y座標のX軸の移動量，およびY軸の移動量を入力してください。

スクリーン座標はワールド座標系に描かれた物体の座標軸に対し，右ネジの回転で前進する方向を正の値とします。

例えば，Xの移動量を50，Yの移動量を30と入力した場合は，平行透視で記述しているので，そのままX＝50，Y＝30を中心に指定することになります。画面上に表示する場合は，そのまま画面右方向に50，上方向に30いった所が中心になります。

次いで，

```
X――ANGLE  =
Y――ANGLE  =
Z――ANGLE  =
```

と聞いてくるので，物体の回転角度を入力してください。

座標記述は，ワールド座標系に描く物体が，座標軸に対し右ネジの回転で前進する方向を正の値とするため，例えば，Xの回転角度を30度，Yを20度，Zを50度と指定した場合には，そのまま，図4－10の矢印の方向にX軸を30度，Y軸を20度，Z軸を50度回転した記述の状態で記憶されます。

次いで，

```
COLOR  =
```

と聞いてくるので，表示したい色番号を黒を入れた8色の中から入力してください。0を指定すると，画面には表示されませんが，実際は黒色で描かれた後のマッピング画面のみ表示されます。

これでスクリーン0画面に物体形状が記述されたので，実際はこの球体にマッピングを施すためのテクスチャパターンの処理画像エリアを指定することになります。コマンドは，

```
SCREEN  X  START  =
SCREEN  X  END    =
SCREEN  Y  START  =
SCREEN  Y  END    =
```

と連続して聞いてくるので，画面エリアを指定してください。ここでテクスチャパターンの処理画像エリアは，スクリーン1の2次元パターンのどの位置でも指定することができます。

例えば，図4—11のように，テクスチャパターン面のマッピングエリアを指定することから，種々の処理が実行されます。この状態で，物体のアングルを変化させている場合は，マッピングされる面が回転しているため，おもしろい結果を生み出します。

マッピングが完了すると，

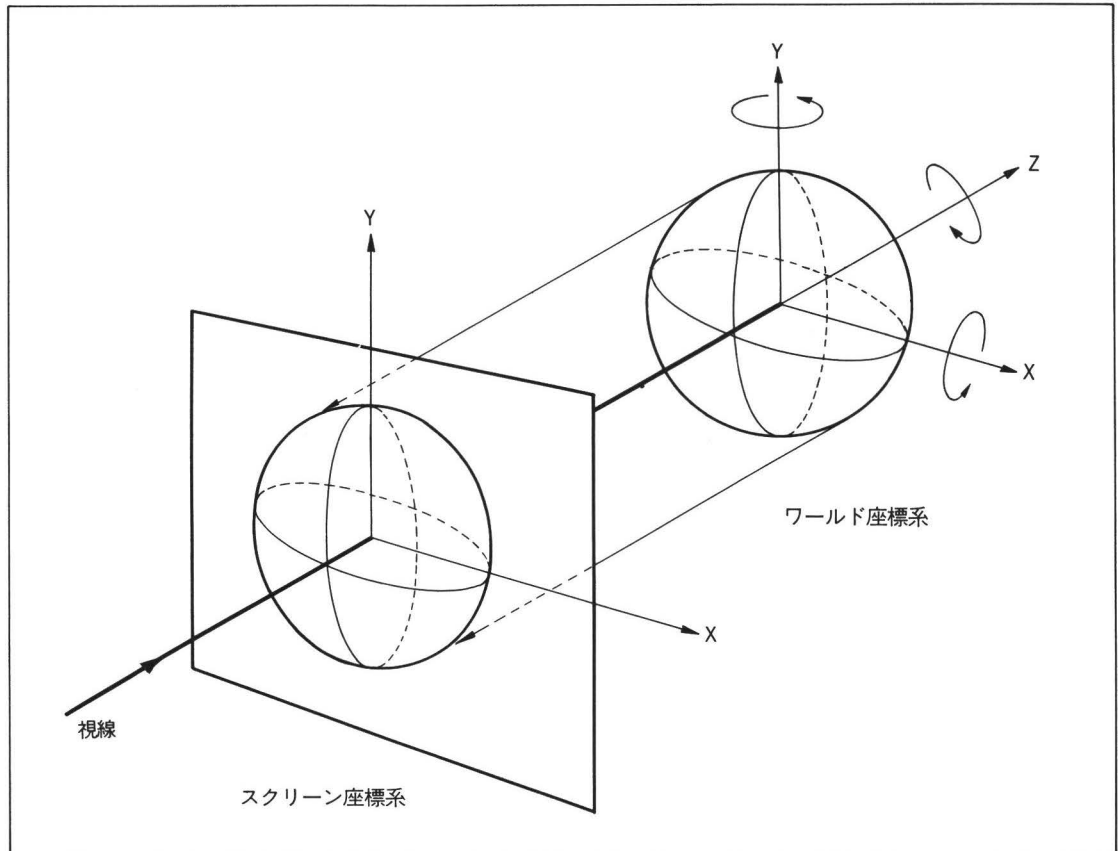



図4—10　ワールド座標／スクリーン座標と視線の関係

```
〔SAVE〕
FILE  NAME  ? ('/'=RETURN)
```

と聞いてくるので，16文字以内でファイル名を付けて SAVE してください。ファイル・ディスクリプタを指定すると，それぞれのディバイスル・ファイルが可能です。

0：フロッピーのドライブ0に記録する

1：フロッピーのドライブ1に記録する

CAS：カセットデータレコーダに記録する

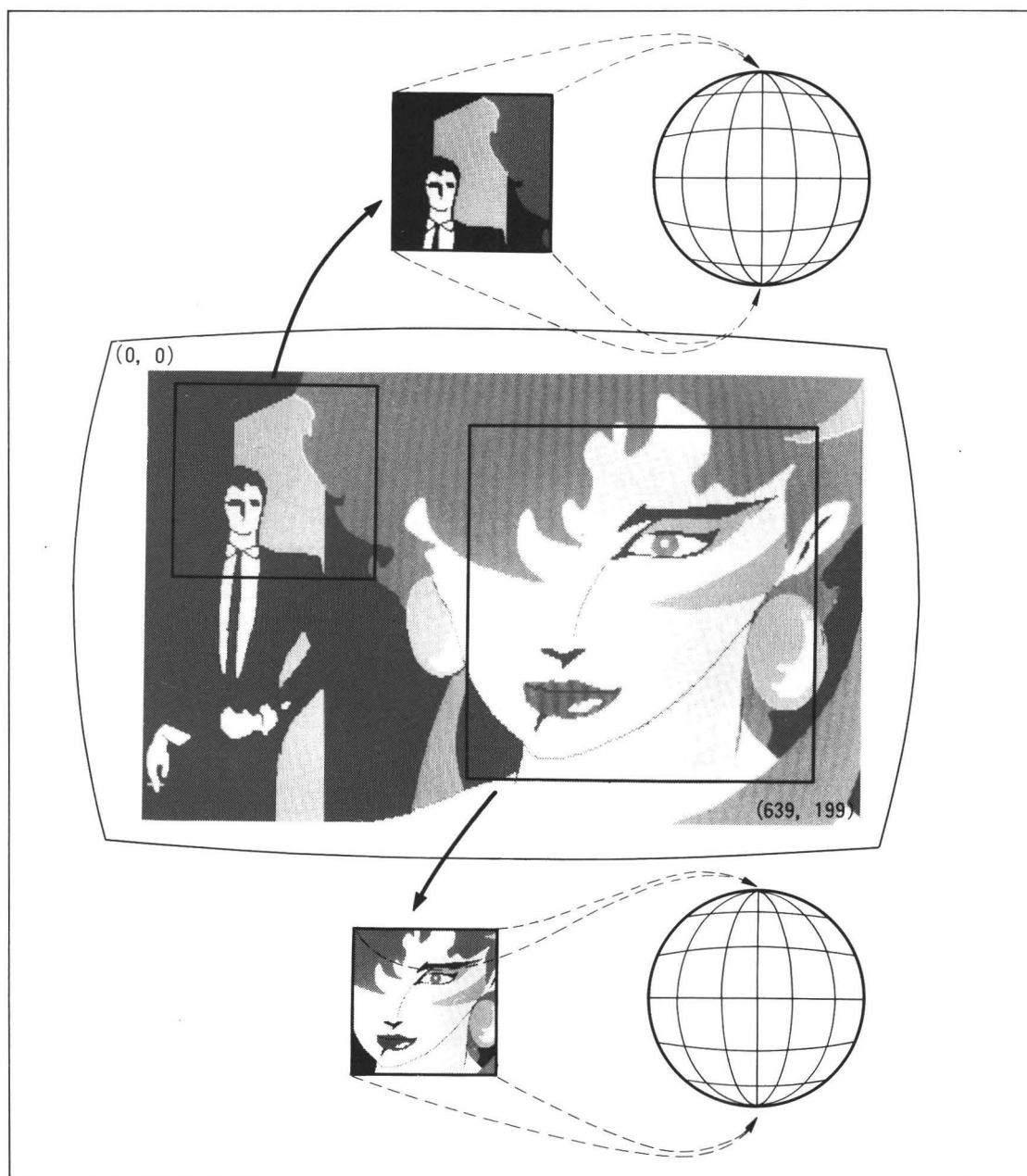
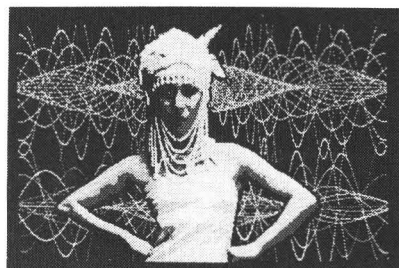
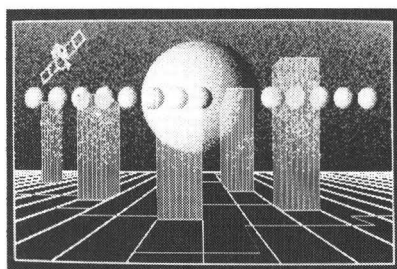
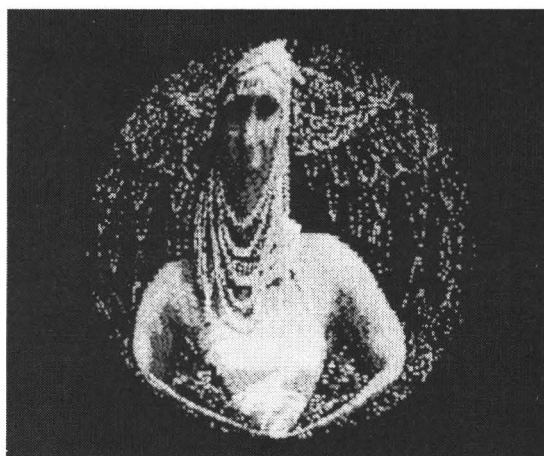
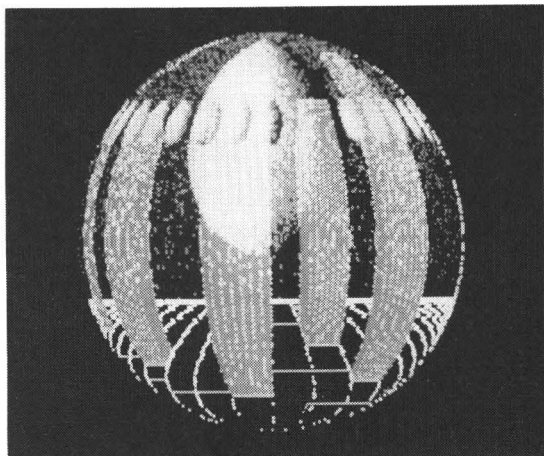



図4—11　マッピングエリアの指定

・球体へのマッピング 出力例

テクスチャパターン

テクスチャパターン

テクスチャパターン

プログラムの内容

プログラムの内容については，球体の形状記述式やマッピング計算方法について，すでに解説済みですので，ここであらためて述べるまでもないと思いますが，フローチャートに基づいて解説します。なお，このプログラムはＸ１ターボ用の640×200ドット画面として書いてあるので，Ｘ１シリーズでご使用の読者は（320×200ドットで使用します），Ｘ１シリーズ用のプログラム変更リストを参考にして入れかえてください。

行番号100から130で初期設定をしています。130行のＸ０,Ｙ０は，スクリーン画面０の座標原点を表しています。原点変更をしたい方は，変化させて違いを見てください。210行で，画面をテクスチャパターン画面（Ｘ１ターボご使用の読者はスクリーン２，Ｘ１シリーズご使用の読者はスクリーン１）に指定した後，4000～4130行でテクスチャパターン画面をLOADします。このテクスチャパターン画面は，前著「パソコングラフィックスの作り方楽しみ方」（学研版）で紹介した「デザインツール」で作成した画面でも，本書第２章で紹介している「デザインツール２」で作成した画面のどちらでも使用することができます。

また，あたりまえのことですが，一度マッピングした画面をあらためてSAVEした後，LOADして使用することも可能です。220行で，画面をマッピング画面（スクリーン０）に切り替えて，1000～1030行のマッピングする物体（球体）の座標記述のサブルーチンにとばし，また，3000～3110行の３次元座標変換値の入力サブルーチンにとばしています。

230行で，球体の描く色が０（黒色）の場合は，描かずにそのまま2000行以下のマッピングのためのサブルーチンに，０以外の場合は，1500～1680行のサブルーチンで球体を描きます。2000～2240行は，マッピングのためのサブルーチンです。詳しくは，すでに計算式の解説をしていますので，フローチャートを参考にしながら解明してください。

マッピングが終了すると，260～300行のファイルのためのメニュー表示に戻ります。310行でキー入力待ちの後，1キーインで6000～6150行の画面SAVEのサブルーチン，2キーインで5000～5160行のLOADのサブルーチン，3キーインで終了となります。

変数とその内容

変数	内容	変数	内容
PAL（ ）	パレットコード	XS	マッピング用画面の左端
XO	スクリーン中央の位置（X座標）	YS	マッピング用画面の上端
YO	スクリーン中央の位置（Y座標）	XE	マッピング用画面の右端
STP	円弧の角度（STEP）	YE	マッピング用画面の下端
STP1		CC	マッピング位置のカラー
PC	カラーコード	MVX	スクリーンX方向の移動
PX	物体の位置（X座標）	MVY	スクリーンY方向の移動
PY	物体の位置（Y座標）	RTX	物体のX軸に対する回転
PZ	物体の位置（Z座標）	RTY	物体のY軸に対する回転
XP	3D変換後のスクリーン（X座標）	RTZ	物体のZ軸に対する回転
YP	3D変換後のスクリーン（Y座標）		

球体マッピング・プログラムの流れ
はじめ
初期設定
100〜 130
画面設定
210
テクスチャパターン
画面LOAD
4000〜4130
画面設定
220
*
マッピング物体(球)の
半径入力
1000〜1030
3D 変換入力
3000〜3110
Y
PC＝0
230
N
球
WIRE FRAME 描く
1500〜1680
マッピングルーチン
2000〜2240
メニュー表示
260〜300
キー入力
310
A$
"1"
6000〜6150
画面SAVE
"2"
5000〜5160
画面LOAD
N
"3"
Y
おわり
マッピングルーチン
マッピング画面範囲
設定
2000〜2080
Y方向　物体と
画面の倍率の決定
2090
FOR（画面Y方向）
2100
物体のY座標計算
2110
X方向　物体と
画面倍率の決定
2120
FOR（画面X方向）
2130
マッピング画から
色を求める
2140〜2160
物体のX座標
Z座標　計算
2170
3D 変換計算
3500〜3580
物体上にマッピング
2190
NEXT
NEXT
RETURN

球体マッピング・プログラムリスト

```
1 '************************************
2 '*                                  *
3 '*     MAPPING - SPHERE       .gak  *
4 '*                                  *
5 '*      Programmed By [ SAIMU ]     *
6 '*                                  *
7 '*          1985 . 4 . 1            *
8 '*                                  *
9 '************************************
100 '---------- ｶﾞﾒﾝ ｾｯﾃｲ ---------------------
110 KLIST 0:OPTION SCREEN 0:WIDTH 80,25,0,0:CLS
120 DIM PAL(7)
130 XO=320:YO=100:STP=10:STP1=30
140 FOR I=0 TO 7:PAL(I)=I:NEXT
200 '---------- ﾒｲﾝ ﾙｰﾁﾝ ---------------------
210 SCREEN 2,2,0:GOSUB 4000
220 SCREEN 0,0,0:GOSUB 1000:GOSUB 3000
230 IF PC=0 THEN 250
240 COLOR PC:GOSUB 1500
250 COLOR 7 :GOSUB 2000
260 PRW 0:CLS:PRINT "[MAPPING FILE ]"
270 PRINT " 1. SCREEN SAVE"
280 PRINT " 2. SCREEN LOAD"
290 PRINT " 3. END"
300 PRINT "   Push Key 1 OR 2 OR 3 ";
310 A$=INKEY$(1)
320 ON INSTR("123",A$) GOSUB 360,370,340
330 GOTO 310
340 INIT:END
350 '-------------------
360 GOSUB 6000:GOTO 260
370 GOSUB 5000:GOTO 260
1000 '---------- ｷｭｳ ﾉ ﾊﾝｹｲ ﾆｭｳﾘｮｸ -------------
1010 CLS
1020 INPUT "ｷｭｳ ﾉ ﾊﾝｹｲ =",R
1030 IF R=<0 THEN 1020 ELSE RETURN
1500 '---------- ｷｭｳ ｦ ｶｸ --------------------
1510 FOR Z=-R TO R STEP STP
1520   R1=SQR(R*R-Z*Z):PX=R1:PY=Z:PZ=0:GOSUB 3500
1530   LINE (XP,YP)-(XP,YP),PSET,PC
1540   FOR TH=0 TO 360 STEP STP
1550     PX=R1*COS(RAD(TH)):PZ=R1*SIN(RAD(TH)):PY=Z
1560     GOSUB 3500:LINE -(XP,YP)
1570   NEXT
1580 NEXT
1590 FOR TH=0 TO 360 STEP STP1
1600   PX=0:PY=-R:PZ=0:GOSUB 3500
```

```
1610    LINE (XP,YP)-(XP,YP),PSET,PC
1620    FOR Z=-R TO R STEP STP
1630      R1=SQR(R*R-Z*Z)
1640      PX=R1*COS(RAD(TH)):PY=Z:PZ=R1*SIN(RAD(TH))
1650      GOSUB 3500:LINE -(XP,YP)
1660    NEXT
1670 NEXT
1680 RETURN
2000 '----------  ﾏｯﾋﾟﾝｸﾞ ﾙｰﾁﾝ  ----------------
2010 LOCATE 0,7
2020 INPUT"SCREEN X START= ",XS
2030 INPUT"SCREEN X END   = ",XE
2040 INPUT"SCREEN Y START= ",YS
2050 INPUT"SCREEN Y END   = ",YE
2060 IF XS<0 OR YS<0 THEN 2000
2070 IF XE>639 OR YE>199 THEN 2000
2080 IF XS>XE OR YS>YE THEN 2000
2090 PRW &HFF:LX=XE-XS:LY=YE-YS:SCY=PAI(R)/LY
2100 IF SCY<1 THEN STP2=1 ELSE STP2=1/SCY
2110 IF SCX<1 THEN STP3=1 ELSE STP3=1/SCX
2120 FOR J=1/SCY TO LY STEP STP2
2130    PY=R*COS(PAI(J/LY))
2140    LL=SQR(R*R-PY*PY):SCX=PAI(LL)/LX
2150    FOR I=0 TO LX STEP STP3
2160      SCREEN 0,2,0
2170      CC=POINT(INT(XS+I+.5),INT(YS+J+.5))
2180      SCREEN 0,0,0
2190      PX=-LL*COS(PAI(I/LX)):PZ=LL*SIN(PAI(I/LX))
2200      GOSUB 3500
2210      PSET (XP,YP,CC):PSET (XP+1,YP,CC)
2220    NEXT
2230 NEXT
2240 PRW 0:RETURN
3000 '----------  3D ｼﾞｮｳｹﾝ ﾆｭｳﾘｮｸ  -------------
3010 INPUT"X -- MOVE  = ",MVX
3020 INPUT"Y -- MOVE  = ",MVY
3030 INPUT"X -- ANGLE = ",RTX
3040 INPUT"Y -- ANGLE = ",RTY
3050 INPUT"Z -- ANGLE = ",RTZ
3060 SS=SIN(RAD(RTX)):CS=COS(RAD(RTY))
3070 SF=SIN(RAD(RTY)):CF=COS(RAD(RTY))
3080 SP=SIN(RAD(RTZ)):CP=COS(RAD(RTZ))
3090 INPUT"   Color   = ",PC
3100 IF (PC<0)+(PC>7) THEN 3090
3110 RETURN
3500 '----------  3D ﾉ ｹｲｻﾝ  -------------------
3510 XP=PX*CF*CP+PY*(SS*SF*CP-CS*SP)
3520 XP=XP+PZ*(CS*SF*CP+SS*SP)
3530 YP=PX*CF*SP+PY*(SS*SF*SP+CS*CP)
```

```
3540 YP=YP+PZ*(CS*SF*SP-SS*CP)
3550 ZP=-PX*SF+PY*SS*CF+PZ*CS*CF
3560 XP=XO+XP*2+MVX
3570 YP=YO-YP-MVY
3580 RETURN
4000 '----------  ｽｸﾘｰﾝ ｶﾞﾒﾝ ﾉ ﾛｰﾄﾞ  --------------
4010 CLS:PRINT"[ SCREEN LOAD ]"
4020 INPUT"  FILE NAME ? ",T$
4030 IF LEN(T$)>16 THEN 4000
4040 OPTION SCREEN 4:PRW &HFF
4050 OPEN"I",1,T$:REC=0
4060 GOSUB 6500:IF WI<>80 THEN 4000
4070 A$=INPUT$(128,1):B$=INPUT$(128,1)
4080 DEVO$"MEM1:",REC,A$,B$:REC=REC+1
4090 IF REC<&HC0 THEN 4070
4100 CLOSE #1:PRW 0:INIT:OPTION SCREEN 0
4110 FOR I=0 TO 7
4120   PALET I,PAL(I)
4130 NEXT:RETURN
5000 '----------  ｶﾞﾒﾝ ﾉ ﾛｰﾄﾞ  --------------------
5010 CLS:PRINT"[ LOAD ]"
5020 INPUT"  FILE NAME ? ('/'=RETURN) ",T$
5030 IF LEN(T$)>16 THEN 5000
5040 IF T$="/" THEN RETURN
5050 GOSUB 5090
5060 FOR I=0 TO 7
5070   PALET I,PAL(I)
5080 NEXT:RETURN
5090 OPTION SCREEN 4:PRW &HFF
5100 OPEN"I",1,T$:REC=0
5110 GOSUB 6500:IF WI<>80 THEN RETURN 5000
5120 A$=INPUT$(128,1):B$=INPUT$(128,1)
5130 DEVO$"MEM:",REC,A$,B$:REC=REC+1
5140 IF REC<&HC0 THEN 5120
5150 CLOSE #1:PRW 0:INIT
5160 OPTION SCREEN 0:CLS:RETURN
6000 '----------  ｶﾞﾒﾝ ﾉ ｾｰﾌﾞ  --------------------
6010 PRW 0:CLS
6020 PRINT"[ SAVE ]"
6030 INPUT"  FILE NAME ? ('/'=RETURN) ",T$
6040 IF LEN(T$)>16 THEN GOTO 6000
6050 IF T$="/" THEN RETURN
6060 OPTION SCREEN 4:PRW &HFF
6070 OPEN"O",1,T$:REC=0
6080 PRINT #1,80
6090 FOR I=0 TO 7
6100   PRINT #1,PAL(I)
6110 NEXT
6120 DEVI$"MEM:",REC,A$,B$:PRINT#1,A$;B$;
```

```
6130 REC=REC+1
6140 IF REC<&HC0 THEN 6120
6150 GOTO 5150
6490 '----------  WIDTH ﾁｪｯｸ  --------------------
6500 INPUT #1,WI
6510 IF WI<>80 THEN 6550
6520 FOR I=0 TO 7
6530   INPUT #1,PAL(I)
6540 NEXT:RETURN
6550 GOSUB 5150
6560 PRINT"WIDTH IS NOT 80 !! (Push Any key) ";
6570 KY$=INKEY$(1)
6580 RETURN
```

X1シリーズ用変更リスト　X1シリーズを使う場合は次の各行を変更・追加します。

```
110 OPTION SCREEN 1:WIDTH 40:CLS
130 XO=160:YO=100:STP=10:STP1=30

210 SCREEN 1,1,0:GOSUB 4000

2070 IF XE>319 OR YE>199 THEN 2000
2160      SCREEN 0,1,0
2210      PSET(XP,YP,CC)

3560 XP=XO+XP+MVX

4040 OPTION SCREEN 2:PRW &HFF
4050 OPEN"I",1,T$:REC=4
4060 GOSUB 6500:IF WI<>40 THEN 4000
4080 DEVO$"MEM0:",REC,A$,B$:REC=REC+1
4085 IF (REC MOD 8)=0 THEN REC=REC+4
4100 CLOSE #1:PRW 0:INIT:OPTION SCREEN 1

5090 OPTION SCREEN 2:PRW &HFF
5110 GOSUB 6500:IF WI<>40 THEN RETURN 5000
5135 IF (REC MOD 4)=0 THEN REC=REC+4
5160 OPTION SCREEN 1:CLS:RETURN

6060 OPTION SCREEN 2:PRW &HFF
6080 PRINT #1,40
6130 REC=REC+1:IF (REC MOD 4)=0 THEN REC=REC+4

6510 IF WI<>40 THEN 6550
6560 PRINT"WIDTH IS NOT 40 !! (Push Any key) ";
```

円柱へのマッピング・プログラム

円柱へのマッピング処理は，テクスチャパターン画面上に描かれたピクセル情報を，円柱のそれぞれの座標値へ変換させます。

処理エリアは，円柱の側面へのマッピングと，上下の円面へのマッピングの2通りに分けられます。

円柱の側面へのマッピングは，テクスチャパターン画面上に描かれたテクスチャから，必要なパターンのX方向の長さをL_X，Y方向の長さをL_Yとすると，マッピング画面上X方向の倍率SC_Xと，Y方向の倍率SC_Yは，次のように書き表すことができます。

$$SC_X=\pi\cdot R/L_X$$

$$SC_Y=H/L_Y$$

テクスチャパターン画面の1点（I, J）に対応する円周上の点（P_X, P_Y, P_Z）は，

$$\begin{cases} P_X=-R\cos\theta_2 \\ P_Y=H-J\cdot H/L_Y \\ P_Z=R\sin\theta_2 \end{cases}$$

（ただし $\theta_2=I/LX\ \pi$（ラジアン））

と書き表すことができます。

上下の円面へのマッピングは，テクスチャパターン画面の1点（I, J）に対応する上下の円面上の点を（P_X, P_Y, P_Z）とすると，P_Yは，

$P_Y=H$ または 0 （ただしHは上面，0は下面）

と書き表せます。

マッピング画面上のY方向の倍率をSC_Yとすると，

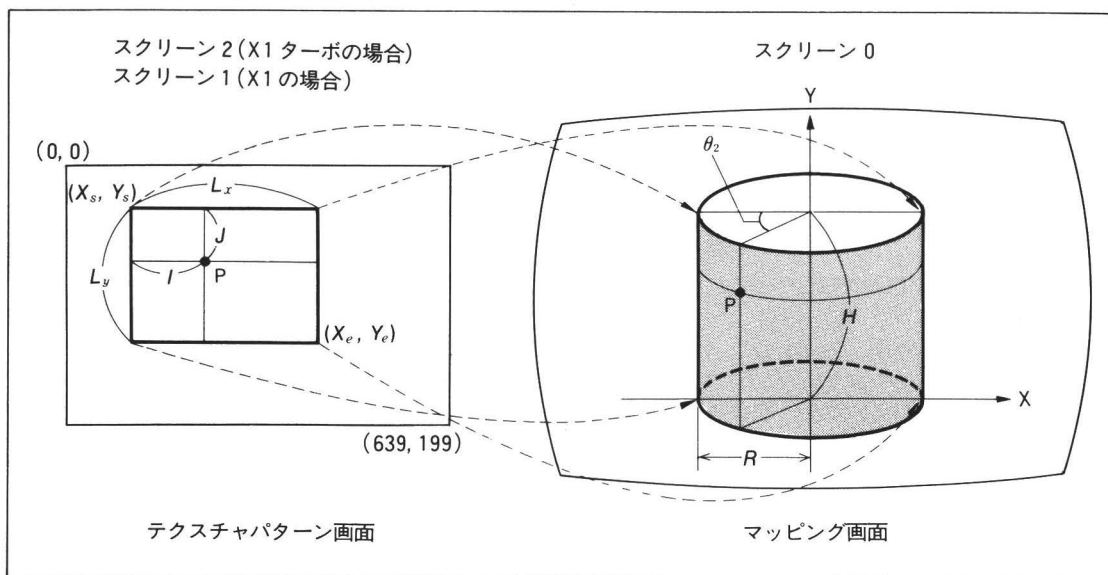



図4—12 円柱の側面へのマッピング

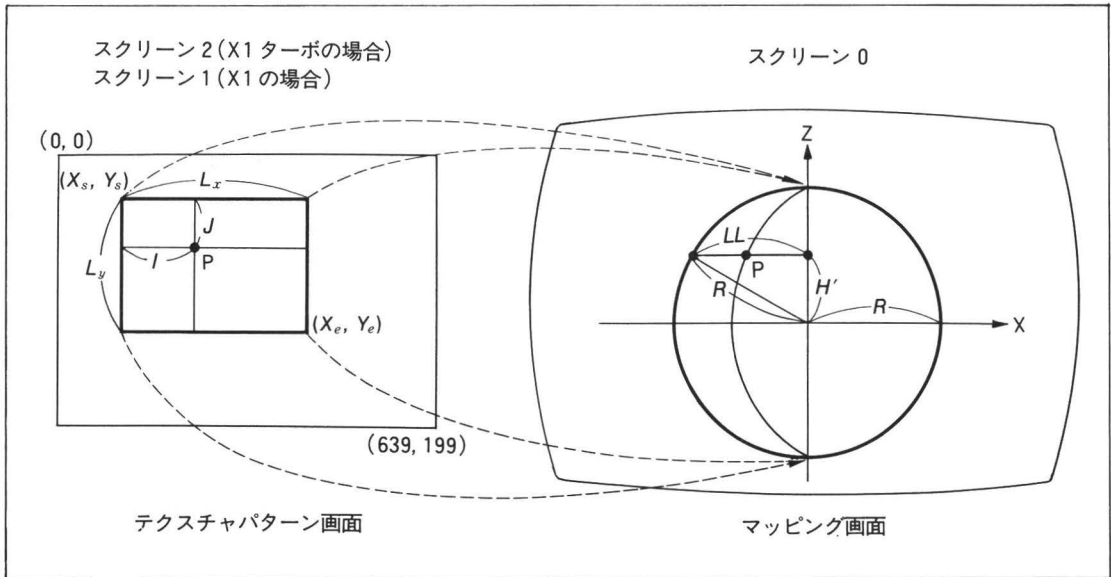



図4—13　円柱の上下の円面へのマッピング

$SC_Y = 2 * R / L_Y$

と書き表すことができます。次に，円弧上の任意の点からZ軸に垂直な線を描き，その長さを LL，Z軸上の線を H' とすると，

$$LL = \sqrt{R^2 - H^2}$$

$$H' = R - J * SCY$$

と書き表せます。マッピング画面のX方向の倍率 SC_X は，

$SC_X = 2LL / L_X$

と表せます。

次いで，テクスチャパターン画面のX方向の点と対応する球の点 P_X, P_Z を求めると

$$\begin{cases} P_X = I \cdot SCX - LL \\ P_Z = J \cdot SCY - R \end{cases}$$

と書き表すことができます。

以上の計算式から，テクスチャパターン上の点 (I, J) の色を決め，P_X, P_Y, P_Z を座標変換したスクリーン上にセットすると，上下の円面へのマッピングが行われます。

プログラムの操作方法

このプログラムを入力して実行（RUN）させると，画面上に，

```
〔SCREEN  LOAD〕
FILE  NAME  ?
```

と聞いてくるので，第2章デザインツール・プログラムで作成した画像のデータのうち，マッピングしたい画像のファイル名を入力して，画面上に読み込んでください。読み込みが完了すると，いよいよマッピング処理のための物体形状の記述に入ります。

画面上には，

```
エントウ ノ R =
エントウ ノ H =
```

と順次聞いてきます。円筒の形状を，半径（R）と高さ（H）の値を決めて入力してください。

次いで，スクリーン座標のどこに描くのかを決定します。球体の場合と同様に，画面のコマンド指示に従って，次の値を入力してください。

```
X——MOVE   =
Y——MOVE   =
X——ANGLE  =
Y——ANGLE  =
Z——ANGLE  =
COLOR  =
```

ただ，円柱の原点は，円柱の底面の中心点にとってあるので，画面中央に描きたい場合は，その位置に注意してください。

次いで，

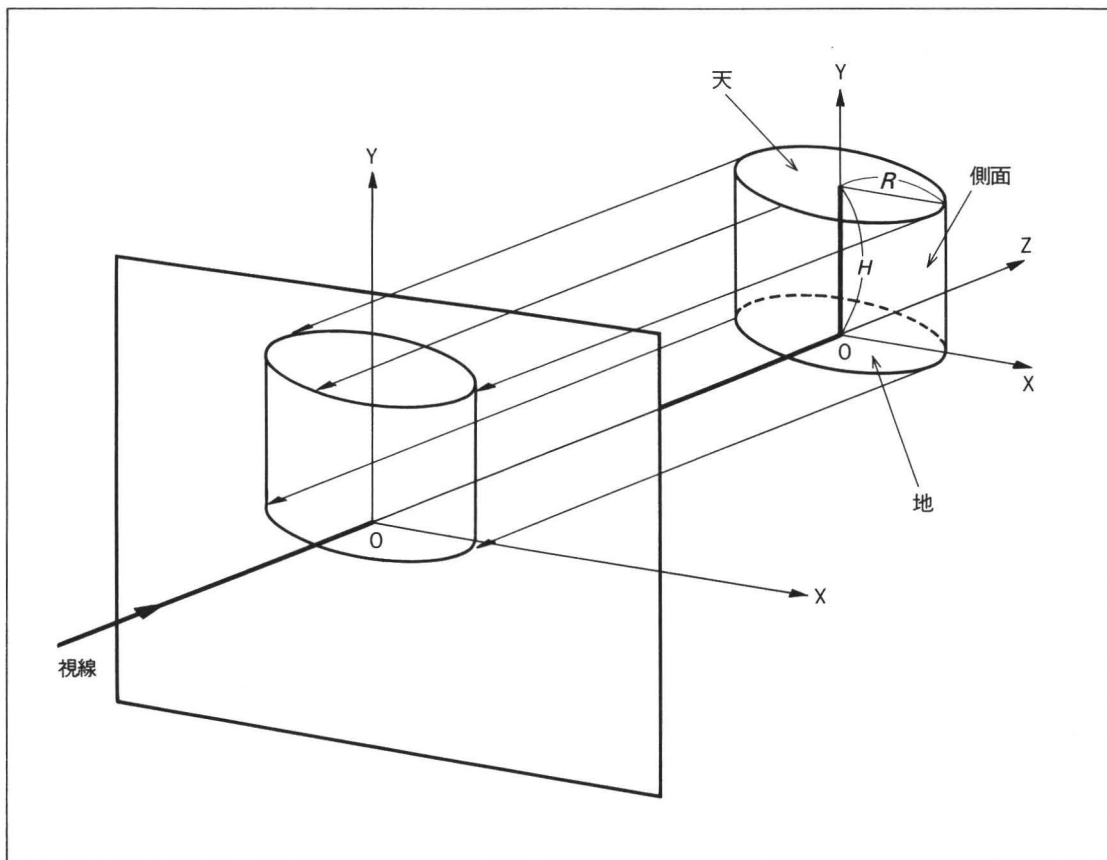



図4—14 ワールド座標／スクリーン座標と視線の関係

```
SCREEN  X  START  =
SCREEN  X  END    =
SCREEN  Y  START  =
SCREEN  Y  END    =
```

と聞いてくるので，テクスチャパターン画面のマッピング表示領域を決めてください（X１ターボの

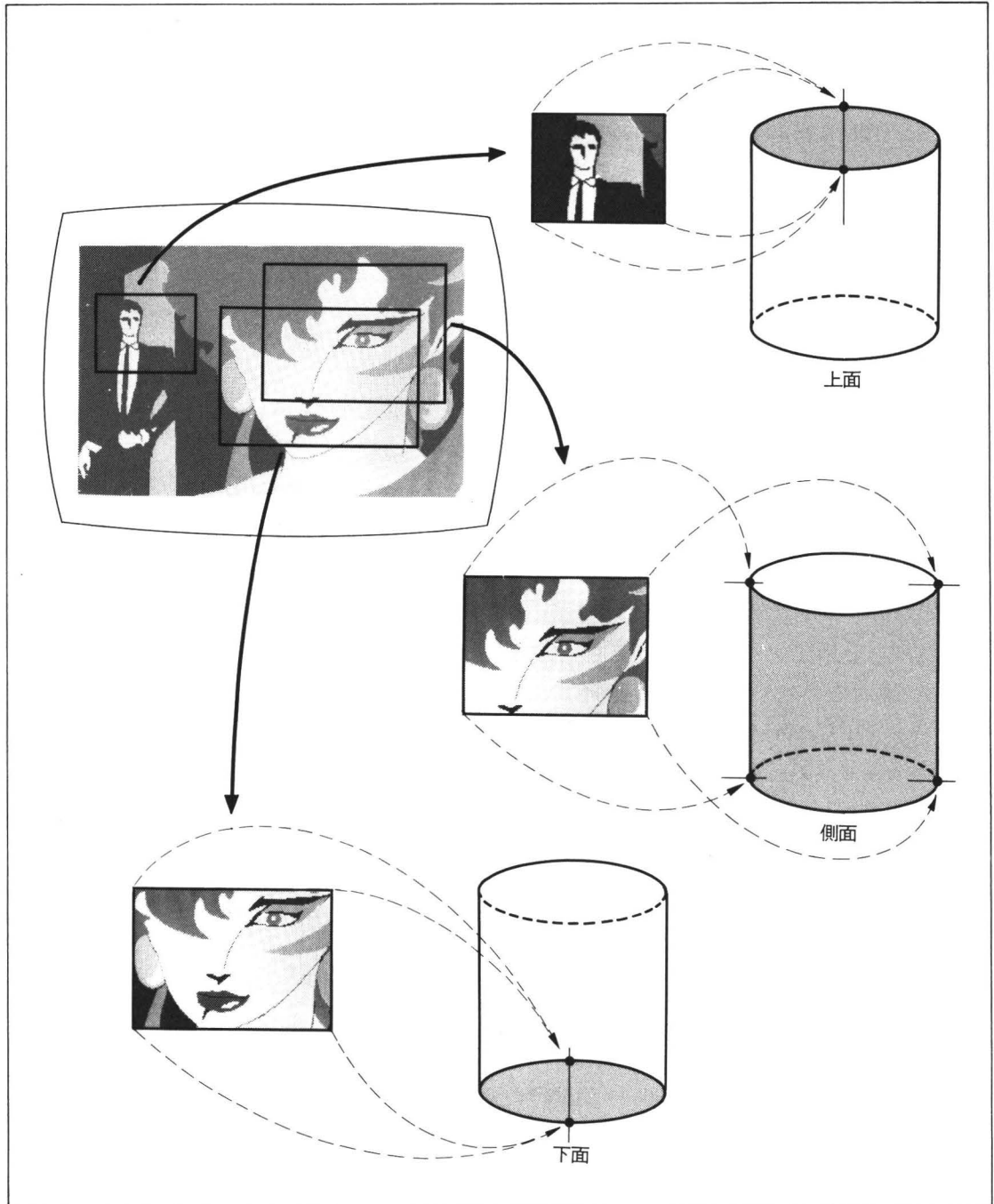



図4－15　マッピングエリアの指定

場合は，Xの値は０～639の間，Yの値は０～199の間で決めてください。Ｘ１の場合は，Xの値は0～319の間，Yの値は0～199の間で決めてください)。

次いで，

```
1. ソクメン
2. テン
3. チ
4. END
```

と聞いてくるので，マッピングしたいエリアの指定を行ってください。

例えば，側面に指定する場合は，[1]をキーインするとマッピングがスタートし，終了すると再びテクスチャパターン画面のマッピング表示領域のコマンドに戻るので，連続して他の画面に処理を行いたい場合は，続けて指定してください。

マッピングが完了すると，

```
〔SAVE〕
FILE  NAME  ? ('/'=RETURN)
```

と聞いてくるので，16文字以内でファイル名を付けて SAVE してください。ファイル・ディスクリプタを指定すると，それぞれのデバイスにファイルが可能です。

プログラムの内容

球体の場合と同様に，Ｘ１シリーズご使用の読者は，Ｘ１シリーズ用のプログラム変更リストと入れかえてご使用ください。

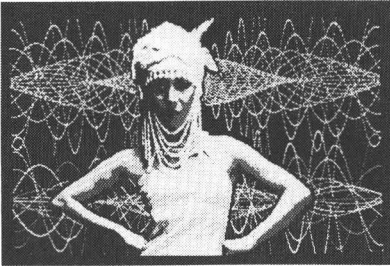
行番号100～130行で初期設定をしています。130行のＸ０，Ｙ０はスクリーン画面０の座標原点を表しています。円柱の座標値計算は，（０，０）点で円柱の下面の中心点にとっています。

210行で画面をテクスチャパターン画面（Ｘ１ターボ使用の読者はスクリーン２，Ｘ１使用の読者はスクリーン１）に指定した後，4000～4130行でテクスチャパターン画面を LOAD します。220行で画面をマッピング画面（スクリーン０）に切り替えて，1000～1050行で円柱の座標記述のサブルーチンにとばし，3000～3110行で３次元の座標変換値の入力サブルーチンにとばしています。

230行で，球体が描く色が０（黒色）の場合は，描かずにそのまま2000行以降のマッピングのためのサブルーチンに，０以外の場合は，1500～1690行のサブルーチンで円柱を描きます。2000～2660行は，マッピングを行うためのサブルーチンです。2200～2350行で円柱の側面へのマッピング，2500～2660行で円柱の上面下面へのマッピングを行っています。これらの計算式は，前述の解説を参考にしてください。

マッピングが終了すると，260～300行のファイルのためのメニュー表示に戻ります。310行でキー入力待ちの後，[1]キーインで6000～6150行の画面 SAVE のサブルーチン，[2]キーインで5000～5160行の LOAD のサブルーチン，[3]キーインで終了となります。

円柱へのマッピング 出力例

テクスチャパターン

テクスチャパターン

筒

変数とその内容			
PAL（　）	パレットコード	XS	マッピング用画面の左端
XO	スクリーン中央の位置（X座標）	YS	マッピング用画面の上端
YO	スクリーン中央の位置（Y座標）	XE	マッピング用画面の右端
STP	円弧の角度（STEP）	YE	マッピング用画面の下端
STP1		CC	マッピング位置のカラー
PC	カラーコード	MVX	スクリーンX方向の移動
PX	物体の位置（X座標）	MVY	スクリーンY方向の移動
PY	物体の位置（Y座標）	RTX	物体のX軸に対する回転
PZ	物体の位置（Z座標）	RTY	物体のY軸に対する回転
XP	3D変換後のスクリーン（X座標）	RTZ	物体のZ軸に対する回転
YP	3D変換後のスクリーン（Y座標）		

円柱マッピング・プログラムリスト

```
1 '***********************************
2 '*                                 *
3 '*     MAPPING - CYLINDER      gak   *
4 '*                                 *
5 '*      Programmed By [ SAIMU ]      *
6 '*                                 *
7 '*          1985 . 4 . 1           *
8 '*                                 *
9 '***********************************
100 '----------  ガメン セッテイ  --------------------
110 KLIST 0:OPTION SCREEN 0:WIDTH 80,25,0,0:CLS
120 DIM PAL(7)
130 XO=320:YO=100:STP=10:STP1=30
140 FOR I=0 TO 7:PAL(I)=I:NEXT
200 '----------  メイン ルーチン  --------------------
210 SCREEN 2,2,0:GOSUB 4000
220 SCREEN 0,0,0:GOSUB 1000:GOSUB 3000
230 IF PC=0 THEN 250
240 COLOR PC:GOSUB 1500
250 COLOR 7 :GOSUB 2000
260 PRW 0:CLS:PRINT "[MAPPING FILE ]"
270 PRINT " 1. SCREEN SAVE"
280 PRINT " 2. SCREEN LOAD"
290 PRINT " 3. END"
300 PRINT "   Push Key 1 OR 2 OR 3 ";
310 A$=INKEY$(1)
320 ON INSTR("123",A$) GOSUB 360,370,340
330 GOTO 310
340 INIT:END
350 '-------------------
360 GOSUB 6000:GOTO 260
370 GOSUB 5000:GOTO 260
1000 '----------  エンチュウ ノ ハンケイ ト タカサ ニュウリョク  ------
1010 CLS
1020 INPUT "エンチュウ ノ R =",R
1030 IF R=<0 THEN 1010
1040 INPUT "エンチュウ ノ H =",H
1050 IF H<=0 THEN 1010 ELSE RETURN
1500 '----------  エンチュウ ヲ エガク  ----------------
1510 PX=R:PY=H:PZ=0:GOSUB 3500
1520 LINE (XP,YP)-(XP,YP),PSET,PC
1530 FOR TH=0 TO 360 STEP STP
1540   PX=R*COS(RAD(TH)):PZ=R*SIN(RAD(TH))
1550   GOSUB 3500:LINE -(XP,YP)
1560 NEXT
1570 PX=R:PY=0:PZ=0:GOSUB 3500
1580 LINE (XP,YP)-(XP,YP),PSET,PC
1590 FOR TH=0 TO 360 STEP STP
```

```
1600    PX=R*COS(RAD(TH)):PZ=R*SIN(RAD(TH))
1610    GOSUB 3500:LINE -(XP,YP)
1620 NEXT
1630 FOR TH=0 TO 360 STEP STP1
1640    PY=H:PX=R*COS(RAD(TH)):PZ=R*SIN(RAD(TH))
1650    GOSUB 3500:LINE(XP,YP)-(XP,YP),PSET,PC
1660    PY=0:GOSUB 3500
1670    LINE -(XP,YP)
1680    NEXT
1690    PRW 0:RETURN
2000 '----------  ｴﾝﾁｭｳ ﾉ ﾏｯﾋﾟﾝｸﾞ  ---------------
2010 LOCATE 0,8
2020 INPUT"SCREEN X START= ",XS
2030 INPUT"SCREEN X END   = ",XE
2040 INPUT"SCREEN Y START= ",YS
2050 INPUT"SCREEN Y END   = ",YE
2060 IF XS<0 OR YS<0 THEN 2000
2070 IF XE>639 OR YE>199 THEN 2000
2080 IF XS>XE OR YS>YE THEN 2000
2090 PRINT:PRINT"MAPPING AREA"
2100 PRINT"  1. ｿｸﾒﾝ"
2110 PRINT"  2. ﾃﾝ  "
2120 PRINT"  3. ﾁ   "
2130 PRINT"  4. END "
2140 A$=INKEY$(1)
2150 IF A$="1" OR A$="2" OR A$="3" THEN 2170
2160 IF A$="4" THEN RETURN ELSE 2140
2170 PRW &HFF:MS=VAL(A$)
2180 ON MS GOSUB 2200,2500,2500
2190 PRW 0:GOTO 2000
2200 '----------  ﾏｯﾋﾟﾝｸﾞ ｴﾝﾁｭｳ ｿｸﾒﾝ  ------------
2210 LX=XE-XS:LY=YE-YS:SCX=PAI(R)/LX:SCY=H/LY
2220 IF SCY<1 THEN STP2=1 ELSE STP2=1/SCY
2230 IF SCX<1 THEN STP3=1 ELSE STP3=1/SCX
2240 FOR J=0 TO LY STEP STP2
2250    PY=H-J*SCY
2260    FOR I=0 TO LX STEP STP3
2270      SCREEN 0,2,0
2280      CC=POINT(INT(XS+I+.5),INT(YS+J+.5))
2290      SCREEN 0,0,0
2300      PX=-R*COS(PAI(I/LX)):PZ=R*SIN(PAI(I/LX))
2310      GOSUB 3500
2320      PSET (XP,YP,CC):PSET (XP+1,YP,CC)
2330    NEXT
2340 NEXT
2350 RETURN
2500 '----------  ﾏｯﾋﾟﾝｸﾞ ｴﾝﾁｭｳ ﾃﾝﾁ  -------------
2510 LX=XE-XS:LY=YE-YS:PY=H*(3-MS):SCY=2*R/LY
2520 IF SCY<1 THEN STP2=1 ELSE STP2=1/SCY
2530 IF SCX<1 THEN STP3=1 ELSE STP3=1/SCX
```

```
2540 FOR J=1/SCY TO LY STEP STP2
2550   LL=SQR(J*SCY*(2*R-J*SCY))
2560   SCX=2*LL/LX:PZ=J*SCY-R
2570   FOR I=0 TO LX STEP STP3
2580     SCREEN 0,2,0
2590     CC=POINT(INT(XS+I+.5),INT(YS+J+.5))
2600     SCREEN 0,0,0
2610     PX=I*SCX-LL
2620     GOSUB 3500
2630     PSET (XP,YP,CC):PSET (XP+1,YP,CC)
2640   NEXT
2650 NEXT
2660 RETURN
3000 '----------- 3D ｼﾞｮｳｹﾝ ﾆｭｳﾘｮｸ ---------------
3010 INPUT"X -- MOVE  = ",MVX
3020 INPUT"Y -- MOVE  = ",MVY
3030 INPUT"X -- ANGLE = ",RTX
3040 INPUT"Y -- ANGLE = ",RTY
3050 INPUT"Z -- ANGLE = ",RTZ
3060 SS=SIN(RAD(RTX)):CS=COS(RAD(RTY))
3070 SF=SIN(RAD(RTY)):CF=COS(RAD(RTY))
3080 SP=SIN(RAD(RTZ)):CP=COS(RAD(RTZ))
3090 INPUT"   Color   = ",PC
3100 IF (PC<0)+(PC>7) THEN 3090
3110 RETURN
3500 '----------- 3D ﾉ ｹｲｻﾝ ---------------------
3510 XP=PX*CF*CP+PY*(SS*SF*CP-CS*SP)
3520 XP=XP+PZ*(CS*SF*CP+SS*SP)
3530 YP=PX*CF*SP+PY*(SS*SF*SP+CS*CP)
3540 YP=YP+PZ*(CS*SF*SP-SS*CP)
3550 ZP=-PX*SF+PY*SS*CF+PZ*CS*CF
3560 XP=XO+XP*2+MVX
3570 YP=YO-YP-MVY
3580 RETURN
4000 '----------- ｽｸﾘｰﾝ ｶﾞﾒﾝ ﾉ ﾛｰﾄﾞ  --------------
4010 CLS:PRINT"[ SCREEN LOAD ]"
4020 INPUT"  FILE NAME ? ",T$
4030 IF LEN(T$)>16 THEN 4000
4040 OPTION SCREEN 4:PRW &HFF
4050 OPEN"I",1,T$:REC=0
4060 GOSUB 6500:IF WI<>80 THEN 4000
4070 A$=INPUT$(128,1):B$=INPUT$(128,1)
4080 DEVO$"MEM1:",REC,A$,B$:REC=REC+1
4090 IF REC<&HC0 THEN 4070
4100 CLOSE #1:PRW 0:INIT:OPTION SCREEN 0
4110 FOR I=0 TO 7
4120   PALET I,PAL(I)
4130 NEXT:RETURN
5000 '----------- ｶﾞﾒﾝ ﾉ ﾛｰﾄﾞ  -------------------
5010 CLS:PRINT"[ LOAD ]"
```

```
5020 INPUT"  FILE NAME ? ('/'=RETURN) ",T$
5030 IF LEN(T$)>16 THEN 5000
5040 IF T$="/" THEN RETURN
5050 GOSUB 5090
5060 FOR I=0 TO 7
5070   PALET I,PAL(I)
5080 NEXT:RETURN
5090 OPTION SCREEN 4:PRW &HFF
5100 OPEN"I",1,T$:REC=0
5110 GOSUB 6500:IF WI<>80 THEN RETURN 5000
5120 A$=INPUT$(128,1):B$=INPUT$(128,1)
5130 DEVO$"MEM:",REC,A$,B$:REC=REC+1
5140 IF REC<&HC0 THEN 5120
5150 CLOSE #1:PRW 0:INIT
5160 OPTION SCREEN 0:CLS:RETURN
6000 '----------  ｶﾞﾒﾝ ﾉ ｾｰﾌﾞ  ------------------
6010 PRW 0:CLS
6020 PRINT"[ SAVE ]"
6030 INPUT"  FILE NAME ? ('/'=RETURN) ",T$
6040 IF LEN(T$)>16 THEN GOTO 6000
6050 IF T$="/" THEN RETURN
6060 OPTION SCREEN 4:PRW &HFF
6070 OPEN"O",1,T$:REC=0
6080 PRINT #1,80
6090 FOR I=0 TO 7
6100   PRINT #1,PAL(I)
6110 NEXT
6120 DEVI$"MEM:",REC,A$,B$:PRINT#1,A$;B$;
6130 REC=REC+1
6140 IF REC<&HC0 THEN 6120
6150 GOTO 5150
6490 '----------  WIDTH ﾁｪｯｸ  -------------------
6500 INPUT #1,WI
6510 IF WI<>80 THEN 6550
6520 FOR I=0 TO 7
6530   INPUT #1,PAL(I)
6540 NEXT:RETURN
6550 GOSUB 5150
6560 PRINT"WIDTH IS NOT 80 !! (Push Any key) ";
6570 KY$=INKEY$(1)
6580 RETURN
```

X1シリーズ用変更リスト	X1シリーズを使う場合は次の各行を変更します。

```
2070 IF XE>319 OR YE>199 THEN 2000

2270      SCREEN 0,1,0
2320      PSET(XP,YP,CC)

2580      SCREEN 0,1,0
2630      PSET(XP,YP,CC)
```

円錐へのマッピング・プログラム

円錐へのマッピング処理は，テクスチャパターン画面上に描かれたピクセル情報を，円錐のそれぞれの座標値へ変換させせます。処理エリアは，円錐の側面と下部の円面の2通りに分けられます。

円錐の側面へのマッピング処理は，テクスチャパターン画面上に描かれたテクスチャから，必要なパターンのX方向の長さを L_X，Y方向の長さを L_Y，円錐の高さをH，PY 軸に垂直な面の切り口の円の半径を LL とすると，マッピング画面上のX方向の倍率 SC_X と，Y方向の倍率 SC_Y は，次のように表せます。

$SC_Y = H / L_Y$

$LL = J \cdot SC_Y \cdot R / H$

$SC_X = \pi LL / L_X$

その結果，テクスチャパターン画面上の1点（I, J）に対応する円周上の点（P_X, P_Y, P_Z）は，次の式で表すことができます。

$$\begin{cases} PX = -LL \cos\theta \\ P_Y = H - J \cdot H / L_Y \\ P_Z = LL \sin\theta \end{cases}$$

ただし $\theta = I / L_X * \pi$（ラジアン）

円錐の底面へのマッピングは，円柱の底面へのマッピングと同様の計算式なので省略します。

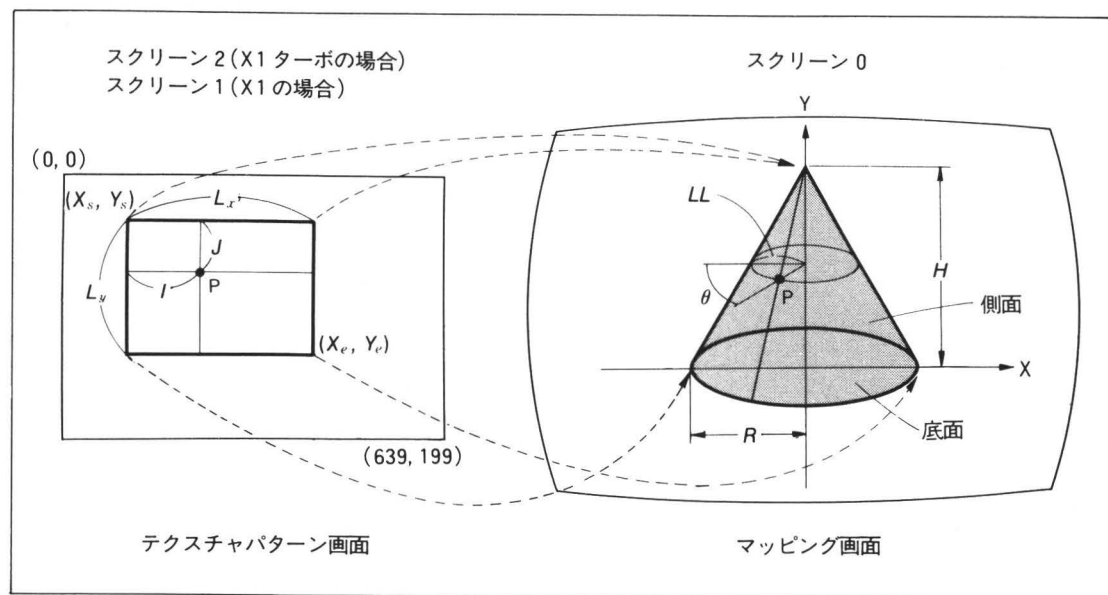



図4—16　円錐の側面へのマッピング

プログラムの操作方法

このプログラムを入力して実行（RUN）させると，画面上に，

```
〔SCREEN　LOAD〕
FILE　NAME　?
```

と聞いてくるので，円柱や円錐へのマッピングと同様に，画像のデータのうちマッピングしたい画像のファイル名を入力してください。読み込みが完了すると，マッピング処理のための物体形状の記述に入ります。

画面上には，

```
エンスイ　ノ　ハンケイ　＝
エンスイ　ノ　タカサ　　＝
```

と順次聞いてくるので，円錐の半径（R）と高さ（H）を入力してください。

次いでスクリーン座標のどこに描くのかを決定します。画面のコマンド指示に従って，次の値を入力してください。

```
X――MOVE　　＝
Y――MOVE　　＝
X――ANGLE　＝
Y――ANGLE　＝
Z――ANGLE　＝
COLOR　　　＝
```

ただし，円錐の原点は，底面の中心点にとってあるので，画面中央に描きたい場合は，その位置に注意してください。

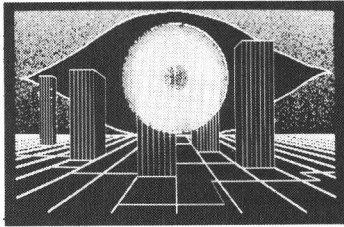
次いで，

```
SCREEN　X　START　＝
SCREEN　X　END　　＝
SCREEN　Y　START　＝
SCREEN　Y　END　　＝
```

と聞いてくるので，テクスチャパターン画面のマッピング表示領域を決めてください（指示領域は円柱や円錐のマッピング・ルーチンと同様です)。

次いで，

```
1. ソクメン
2. テイメン
3. END
```

円錐へのマッピング 出力例

テクスチャパターン

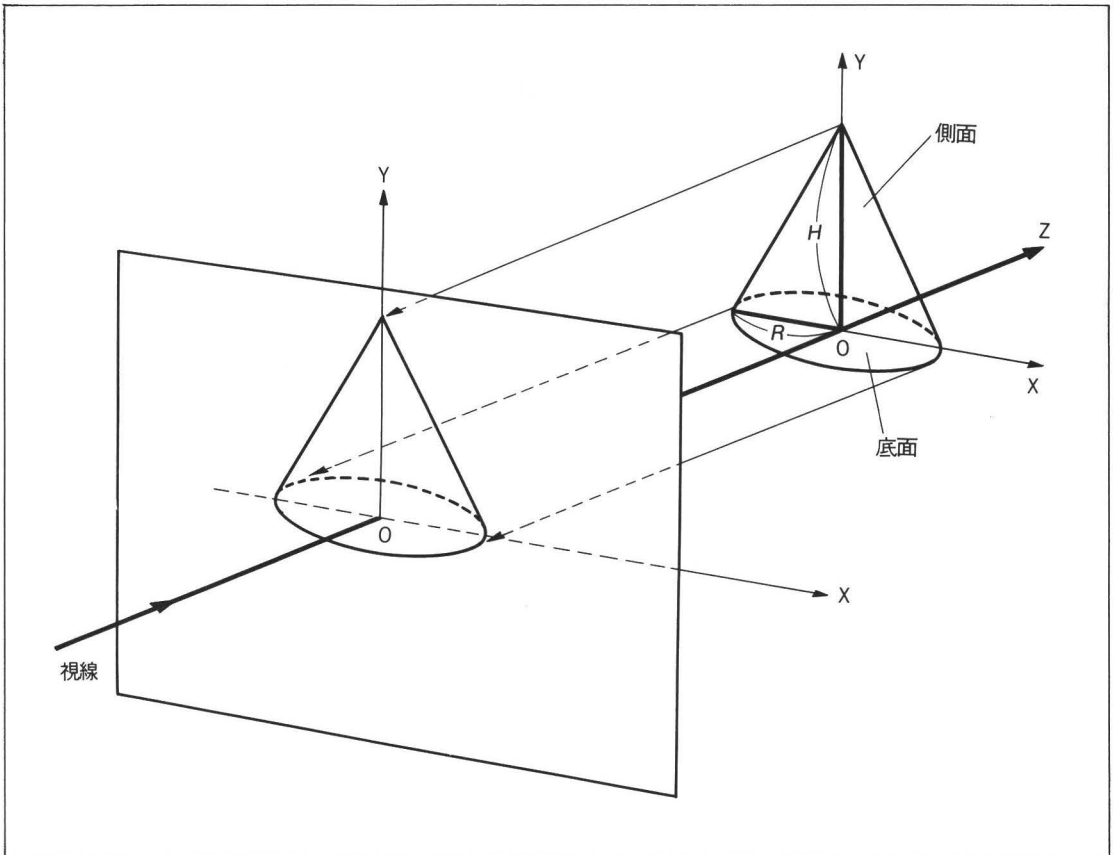
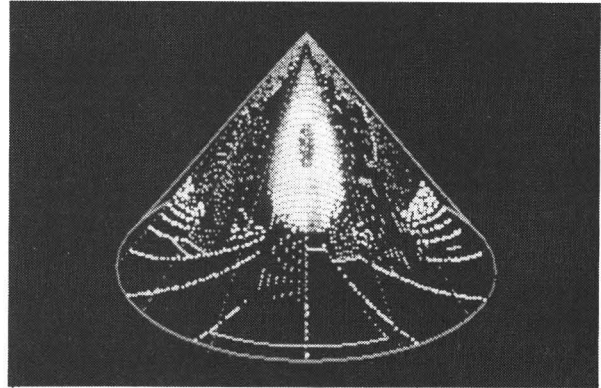



図4—17　ワールド座標／スクリーン座標と視線の関係

と聞いてくるので，マッピングしたいエリアの指定を行ってください。

例えば，側面に指定する場合は[1]をキーインするとマッピングがスタートし，終了すると再びテクスチャパターン画面のマッピング表示領域のコマンドに戻るので，連続して他の画面に処理を行いたい場合は，続けて指定してください。

マッピングが完了すると，

```
〔SAVE〕
EILE  NAME  ?  ('/'=RETURN)
```

と聞いてくるので，16文字以内でファイル名を付けて SAVE してください。ファイル・ディスクリプタを指定すると，それぞれのデバイスにファイルが可能です。

プログラムの内容

プログラムの組み立ては，116ページの円柱の場合とほぼ同じなので，参照してください。

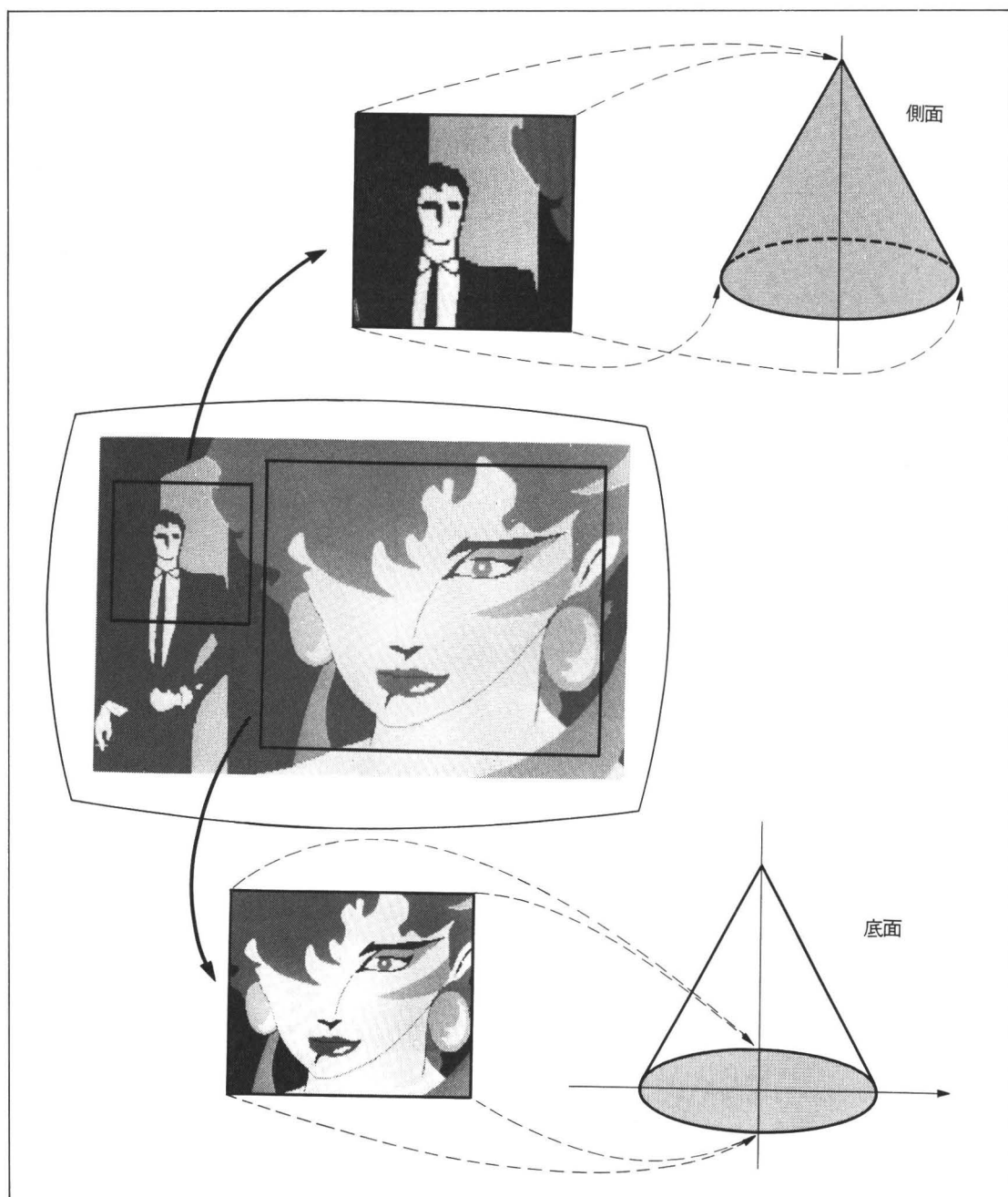



図4—18　マッピングエリアの指定

変数とその内容			
PAL（ ）	パレットコード	XS	マッピング用画面の左端
XO	スクリーン中央の位置（X座標）	YS	マッピング用画面の上端
YO	スクリーン中央の位置（Y座標）	XE	マッピング用画面の右端
STP	円弧の角度（STEP）	YE	マッピング用画面の下端
STP1		CC	マッピング位置のカラー
PC	カラーコード	MVX	スクリーンX方向の移動
PX	物体の位置（X座標）	MVY	スクリーンY方向の移動
PY	物体の位置（Y座標）	RTX	物体のX軸に対する回転
PZ	物体の位置（Z座標）	RTY	物体のY軸に対する回転
XP	3D変換後のスクリーン（X座標）	RTZ	物体のZ軸に対する回転
YP	3D変換後のスクリーン（Y座標）		

円錐マッピング・プログラムの流れ

円錐マッピング・プログラムリスト

```
1 '*************************************
2 '*                                   *
3 '*      MAPPING - CONE         gak   *
4 '*                                   *
5 '*       Programmed By [ SAIMU ]     *
6 '*                                   *
7 '*            1985 . 4 . 1           *
8 '*                                   *
9 '*************************************
100 '---------- ｶﾞﾒﾝ ｾｯﾃｲ ----------------------
110 KLIST 0:OPTION SCREEN 0:WIDTH 80,25,0,0:CLS
120 DIM PAL(7)
130 XO=320:YO=100:STP=10:STP1=30
140 FOR I=0 TO 7:PAL(I)=I:NEXT
200 '---------- ﾒｲﾝ ﾙｰﾁﾝ ----------------------
210 SCREEN 2,2,0:GOSUB 4000
220 SCREEN 0,0,0:GOSUB 1000:GOSUB 3000
230 IF PC=0 THEN 250
240 COLOR PC:GOSUB 1500
250 COLOR 7 :GOSUB 2000
260 PRW 0:CLS:PRINT "[MAPPING FILE ]"
270 PRINT " 1. SCREEN SAVE"
280 PRINT " 2. SCREEN LOAD"
290 PRINT " 3. END"
300 PRINT "   Push Key 1 OR 2 OR 3 ";
310 A$=INKEY$(1)
320 ON INSTR("123",A$) GOSUB 360,370,340
330 GOTO 310
340 INIT:END
350 '------------------
360 GOSUB 6000:GOTO 260
370 GOSUB 5000:GOTO 260
1000 '---------- ｴﾝｽｲ ﾉ ﾊﾝｹｲ ﾀｶｻ ﾆｭｳﾘｮｸ ---------
1010 CLS
1020 INPUT "ｴﾝｽｲ ﾉ ﾊﾝｹｲ = ",R
1030 IF R<=0 THEN 1000
1040 INPUT "ｴﾝｽｲ ﾉ ﾀｶｻ  = ",H
1050 IF H<=0 THEN 1040 ELSE RETURN
1500 '---------- ｴﾝｽｲ ｦ ｴｶﾞｸ ------------------
1510 PRW &HFF
1520 PX=R:PY=0:PZ=0:GOSUB 3500
1530 LINE (XP,YP)-(XP,YP),PSET,PC
1540 FOR TH=0 TO 360 STEP STP
1550   PX=R*COS(RAD(TH)):PZ=R*SIN(RAD(TH))
1560   GOSUB 3500:LINE -(XP,YP)
1570 NEXT
1600 FOR TH=0 TO 360 STEP STP1
1610   PX=0:PY=H:PZ=0:GOSUB 3500
```

```
1620    LINE (XP,YP)-(XP,YP),PSET,PC
1630    PX=R*COS(RAD(TH)):PY=0:PZ=R*SIN(RAD(TH))
1640    GOSUB 3500:LINE -(XP,YP)
1650 NEXT
1660 PRW 0:RETURN
2000 '----------  ﾏｯﾋﾟﾝｸﾞ ﾙｰﾁﾝ  ------------------
2010 LOCATE 0,9
2020 INPUT"SCREEN X START= ",XS
2030 INPUT"SCREEN X END   = ",XE
2040 INPUT"SCREEN Y START= ",YS
2050 INPUT"SCREEN Y END   = ",YE
2060 IF XS<0 OR YS<0 THEN 2000
2070 IF XE>639 OR YE>199 THEN 2000
2080 IF XS>XE OR YS>YE THEN 2000
2090 LX=XE-XS:LY=YE-YS
2100 PRINT:PRINT"MAPPING AREA"
2110 PRINT"  1. ｿｸﾒﾝ"
2120 PRINT"  2. ﾃｲﾒﾝ "
2130 PRINT"  3. END  "
2140 A$=INKEY$(1)
2150 IF A$="1" OR A$="2" THEN 2170
2160 IF A$="3" THEN RETURN ELSE 2140
2170 PRW &HFF:MS=VAL(A$)
2180 ON MS GOSUB 2200,2500
2190 PRW 0:GOTO 2000
2200 '----------  ﾏｯﾋﾟﾝｸﾞ ｴﾝｽｲ (ｿｸﾒﾝ)  -----------
2210 PRW &HFF
2220 SCY=H/LY
2230 IF SCY<1 THEN STP2=1 ELSE STP2=1/SCY
2240 IF SCX<1 THEN STP3=1 ELSE STP3=1/SCX
2250 FOR J=1/SCY TO LY STEP STP2
2260    PY=H-J*SCY:LL=J*SCY*R/H:SCX=PAI(LL)/LX
2270    FOR I=0 TO LX STEP STP3
2280      SCREEN 0,2,0
2290      CC=POINT(INT(XS+I+.5),INT(YS+J+.5))
2300      SCREEN 0,0,0
2310      PX=-LL*COS(PAI(I/LX)):PZ=LL*SIN(PAI(I/LX))
2320      GOSUB 3500
2330      PSET (XP,YP,CC):PSET (XP+1,YP,CC)
2340    NEXT
2350 NEXT
2360 RETURN
2500 '----------  ﾏｯﾋﾟﾝｸﾞ ｴﾝｽｲ (ﾃｲﾒﾝ)  -----------
2510 PRW &HFF
2520 SCY=2*R/LY:PY=0
2530 IF SCY<1 THEN STP2=1 ELSE STP2=1/SCY
2540 IF SCX<1 THEN STP3=1 ELSE STP3=1/SCX
2550 FOR J=1/SCY TO LY STEP STP2
2560    PX=R-J*SCY:R1=SQR(R*R-PX*PX)
2570    SCX=2*R1/LX
```

```
2580    FOR I=0 TO LX STEP STP3
2590      SCREEN 0,2,0
2600      CC=POINT(INT(XS+I+.5),INT(YS+J+.5))
2610      SCREEN 0,0,0
2620      PZ=R1-I*SCX:GOSUB 3500
2630      PSET (XP,YP,CC):PSET (XP+1,YP,CC)
2640    NEXT
2650 NEXT
2660 RETURN
3000 '----------- 3D ジョウケン ニュウリョク ---------------
3010 INPUT"X -- MOVE  = ",MVX
3020 INPUT"Y -- MOVE  = ",MVY
3030 INPUT"X -- ANGLE = ",RTX
3040 INPUT"Y -- ANGLE = ",RTY
3050 INPUT"Z -- ANGLE = ",RTZ
3060 SS=SIN(RAD(RTX)):CS=COS(RAD(RTY))
3070 SF=SIN(RAD(RTY)):CF=COS(RAD(RTY))
3080 SP=SIN(RAD(RTZ)):CP=COS(RAD(RTZ))
3090 INPUT"   Color   = ",PC
3100 IF (PC<0)+(PC>7) THEN 3090
3110 RETURN
3500 '----------- 3D ノ ケイサン ---------------------
3510 XP=PX*CF*CP+PY*(SS*SF*CP-CS*SP)
3520 XP=XP+PZ*(CS*SF*CP+SS*SP)
3530 YP=PX*CF*SP+PY*(SS*SF*SP+CS*CP)
3540 YP=YP+PZ*(CS*SF*SP-SS*CP)
3550 ZP=-PX*SF+PY*SS*CF+PZ*CS*CF
3560 XP=XO+XP*2+MVX
3570 YP=YO-YP-MVY
3580 RETURN
4000 '----------- スクリーン ガメン ノ ロード  --------------
4010 CLS:PRINT"[ SCREEN LOAD ]"
4020 INPUT"  FILE NAME ? ",T$
4030 IF LEN(T$)>16 THEN 4000
4040 OPTION SCREEN 4:PRW &HFF
4050 OPEN"I",1,T$:REC=0
4060 GOSUB 6500:IF WI<>80 THEN 4000
4070 A$=INPUT$(128,1):B$=INPUT$(128,1)
4080 DEVO$"MEM1:",REC,A$,B$:REC=REC+1
4090 IF REC<&HC0 THEN 4070
4100 CLOSE #1:PRW 0:INIT:OPTION SCREEN 0
4110 FOR I=0 TO 7
4120   PALET I,PAL(I)
4130 NEXT:RETURN
5000 '----------- ガメン ノ ロード  ------------------
5010 CLS:PRINT"[ LOAD ]"
5020 INPUT"  FILE NAME ? ('/'=RETURN) ",T$
5030 IF LEN(T$)>16 THEN 5000
5040 IF T$="/" THEN RETURN
5050 GOSUB 5090
```

```
5060 FOR I=0 TO 7
5070   PALET I,PAL(I)
5080 NEXT:RETURN
5090 OPTION SCREEN 4:PRW &HFF
5100 OPEN"I",1,T$:REC=0
5110 GOSUB 6500:IF WI<>80 THEN RETURN 5000
5120 A$=INPUT$(128,1):B$=INPUT$(128,1)
5130 DEVO$"MEM:",REC,A$,B$:REC=REC+1
5140 IF REC<&HC0 THEN 5120
5150 CLOSE #1:PRW 0:INIT
5160 OPTION SCREEN 0:CLS:RETURN
6000 '----------  ｶﾞﾒﾝ ﾉ ｾｰﾌﾞ  ------------------
6010 PRW 0:CLS
6020 PRINT"[ SAVE ]"
6030 INPUT"  FILE NAME ? ('/'=RETURN) ",T$
6040 IF LEN(T$)>16 THEN GOTO 6000
6050 IF T$="/" THEN RETURN
6060 OPTION SCREEN 4:PRW &HFF
6070 OPEN"O",1,T$:REC=0
6080 PRINT #1,80
6090 FOR I=0 TO 7
6100   PRINT #1,PAL(I)
6110 NEXT
6120 DEVI$"MEM:",REC,A$,B$:PRINT#1,A$;B$;
6130 REC=REC+1
6140 IF REC<&HC0 THEN 6120
6150 GOTO 5150
6490 '----------  WIDTH ﾁｪｯｸ  ------------------
6500 INPUT #1,WI
6510 IF WI<>80 THEN 6550
6520 FOR I=0 TO 7
6530   INPUT #1,PAL(I)
6540 NEXT:RETURN
6550 GOSUB 5150
6560 PRINT"WIDTH IS NOT 80 !! (Push Any key) ";
6570 KY$=INKEY$(1)
6580 RETURN
```

X1シリーズ用変更リスト　X1シリーズを使う場合は次の各行を変更します。

```
2070 IF XE>319 OR YE>199 THEN 2000

2280      SCREEN 0,1,0
2330      PSET(XP,YP,CC)

2590      SCREEN 0,1,0
2630      PSET (XP,YP,CC)
```

直方体へのマッピング・プログラム

直方体へのマッピング処理は，テクスチャパターン画面上に描かれたピクセル情報を，直方体のそれぞれの座標値へ変換させます。処理エリアとしては，直方体の正面，立面，側面の3通りに分けられます。

直方体のマッピング処理は，テクスチャパターン画面上に描かれたテクスチャから，必要なパターンをそれぞれ直方体の正面，立面，側面の各四角形に対応させて表します。

テクスチャパターン画面上に描かれたテクスチャから，必要なパターンのX方向の長さを L_X，Y方向の長さを L_Y，直方体の縦の長さをH，横の長さをW，奥行をD，マッピング画面上のX方向の倍率を SC_X，Y方向の倍率を SC_Y，テクスチャパターン画面上の1点（I, J）に対応する面上の点（P_X, P_Y, P_Z）はそれぞれ次のように表現することができます。

●正面へのマッピングの場合

$$\begin{cases} SC_X = W / L_X \\ SC_Y = H / L_Y \end{cases}$$

これにより，次の式が導かれます。

$$\begin{cases} P_X = I \cdot SC_X \\ P_Y = H - J \cdot SC_Y \\ P_Z = D \end{cases}$$

●立面へのマッピングの場合

$$\begin{cases} SC_X = W / L_X \\ SC_Y = D / L_Y \end{cases}$$

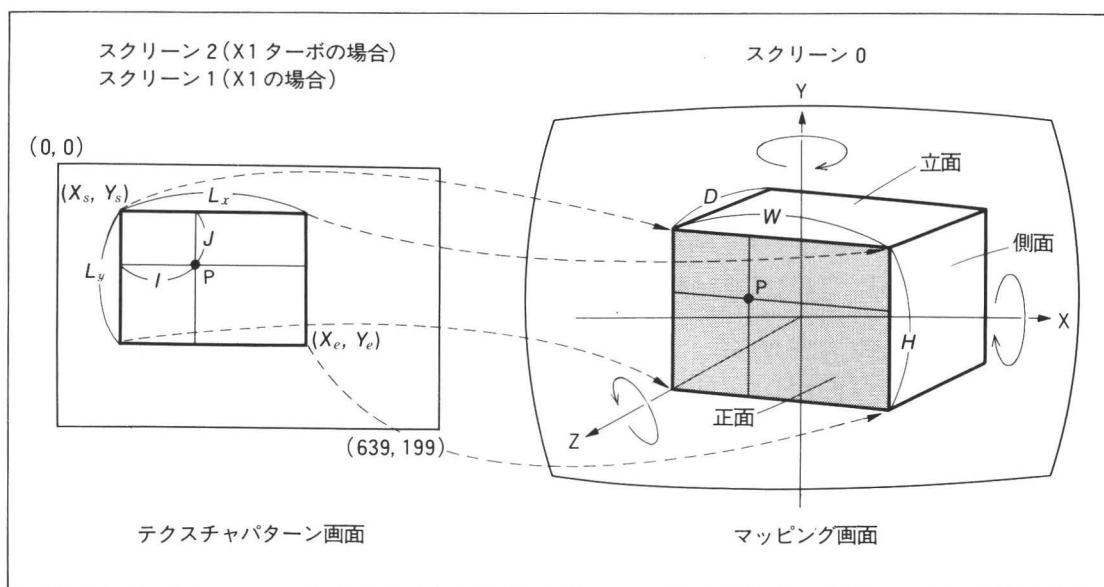



図4―19　直方体へのマッピング(正面への例)

これにより，次の式が求められます。

$$\begin{cases} P_X = I \cdot SC_X \\ P_Y = H \\ P_Z = J \cdot SC_Y \end{cases}$$

●側面へのマッピングの場合

$$\begin{cases} SC_X = D / L_X \\ SC_Y = H / L_Y \end{cases}$$

これにより，次のように表すことができます。

$$\begin{cases} P_X = W \\ P_Y = H - D \cdot SC_Y \\ P_Z = D - I \cdot SC_X \end{cases}$$

プログラムの操作方法

このプログラムを入力して実行（RUN）させると，画面上に，

```
〔SCREEN　LOAD〕
FILE　NAME　?
```

と聞いてくるので，円柱等のマッピング・プログラムと同様，画像データのうちマッピングしたい画像のファイル名を入力してください。

読み込みが完了すると，マッピング処理のための物体形状の記述に入ります。

画面上には，

```
タテノ　ナガサ　＝
ヨコノ　ナガサ　＝
タカサ　　　　　＝
```

と順次聞いてくるので，直方体の縦の長さ（D），横の長さ（W），高さ（H）を入力してください（この場合は，直方体は絶対座標系の正方向の3面だけ描くようにしてあるので注意してください）。

次いで，スクリーン座標のどこに描くのか決定します。円柱等のマッピング・ルーチンと同様に画面のコマンド指示に従って，次の値を入力してください。

```
X――MOVE　 ＝
Y――MOVE　 ＝
X――ANGLE ＝
Y――ANGLE ＝
Z――ANGLE ＝
COLOR　　 ＝
```

次いで，

```
SCREEN X START =
SCREEN X END   =
SCREEN Y START =
SCREEN Y END   =
```

と聞いてくるので，テクスチャパターン画面のマッピング表示領域を決めてください（指示領域は他のマッピングルーチンと同様です）。

次いで，

```
1. ショウメン
2. リツメン
3. ソクメン
4. END
```

と聞いてくるので，マッピングしたいエリアの指定を行ってください。

例えば，正面に指定する場合は，[1]をキーインするとマッピングがスタートし，終了すると再びテクスチャパターン画面のマッピング表示領域のコマンドに戻るので，連続して他の画面に処理を行いたい場合は，続けて指定してください。

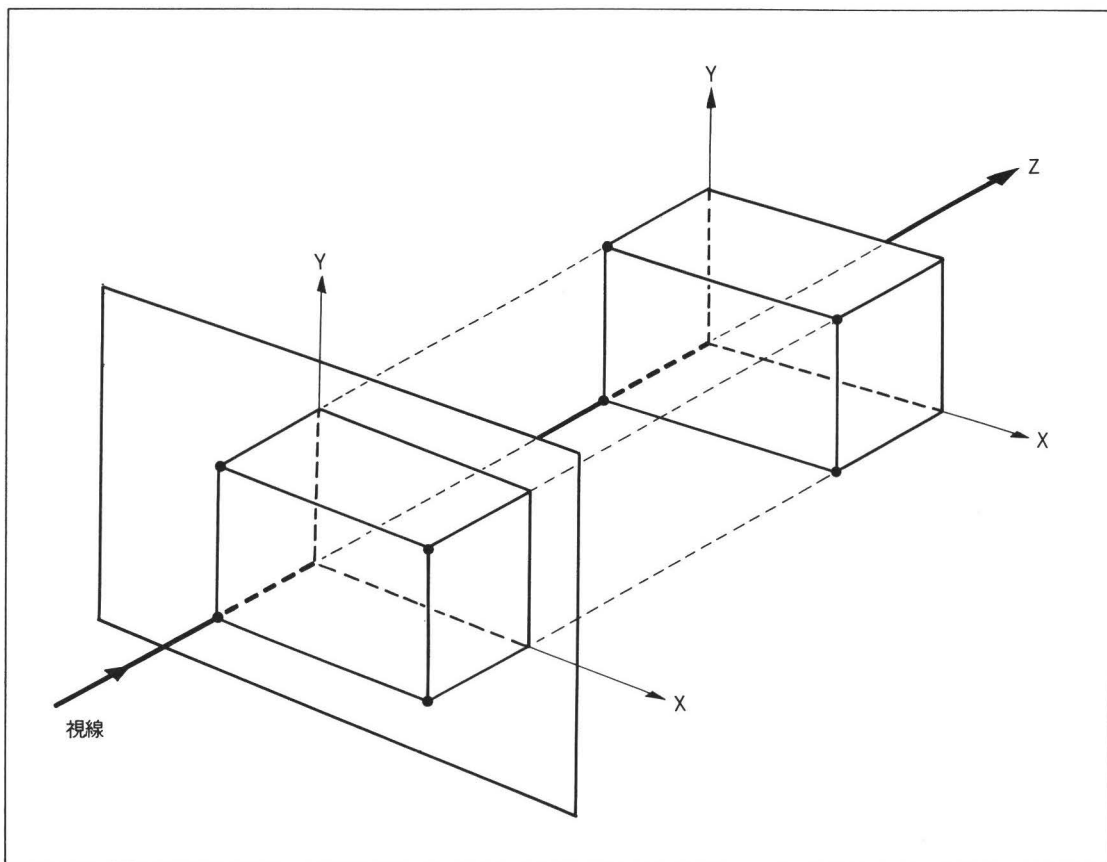



図4—20　ワールド座標／スクリーン座標と視線の関係

マッピングが完了すると，

```
〔SAVE〕
FILE  NAME  ?  ('/'=RETURN)
```

と聞いてくるので，16文字以内でファイル名を付けて SAVE してください。ファイル・ディスクリプタを指定すると，それぞれのデバイスにファイルが可能です。

プログラムの内容

プログラムの組み立ては，116ページの円柱の場合とほぼ同じなので，参照してください。

図4―21　マッピングエリアの指定

直方体へのマッピング 出力例

テクスチャパターン

テクスチャパターン

テクスチャパターン

変数とその内容

変数	内容	変数	内容
PAL（　）	パレットコード	XS	マッピング用画面の左端
XO	スクリーン中央の位置（X座標）	YS	マッピング用画面の上端
YO	スクリーン中央の位置（Y座標）	XE	マッピング用画面の右端
STP	円弧の角度（STEP）	YE	マッピング用画面の下端
STP1	円弧の角度（STEP）	CC	マッピング位置のカラー
PC	カラーコード	MVX	スクリーンX方向の移動
PX	物体の位置（X座標）	MVY	スクリーンY方向の移動
PY	物体の位置（Y座標）	RTX	物体のX軸に対する回転
PZ	物体の位置（Z座標）	RTY	物体のY軸に対する回転
XP	3D変換後のスクリーン（X座標）	RTZ	物体のZ軸に対する回転
YP	3D変換後のスクリーン（Y座標）		

直方体マッピング・プログラムの流れ

直方体マッピング・プログラムリスト

```
1 '*************************************
2 '*                                   *
3 '*     MAPPING - PLANE         gak   *
4 '*                                   *
5 '*       Programmed By [ SAIMU ]     *
6 '*                                   *
7 '*           1985 . 4 . 1            *
8 '*                                   *
9 '*************************************
100 '----------- ガメン セッテイ -----------------------
110 KLIST 0:OPTION SCREEN 0:WIDTH 80,25,0,0:CLS
120 DIM PAL(7)
130 XO=320:YO=100:STP=10:STP1=30
140 FOR I=0 TO 7:PAL(I)=I:NEXT
200 '----------- メイン ルーチン -----------------------
210 SCREEN 2,2,0:GOSUB 4000
220 SCREEN 0,0,0:GOSUB 1000:GOSUB 3000
230 IF PC=0 THEN 250
240 COLOR PC:GOSUB 1500
250 COLOR 7 :GOSUB 2000
260 PRW 0:CLS:PRINT "[MAPPING FILE ]"
270 PRINT " 1. SCREEN SAVE"
280 PRINT " 2. SCREEN LOAD"
290 PRINT " 3. END"
300 PRINT "   Push Key 1 OR 2 OR 3 ";
310 A$=INKEY$(1)
320 ON INSTR("123",A$) GOSUB 360,370,340
330 GOTO 310
340 INIT:END
350 '---------------------
360 GOSUB 6000:GOTO 260
370 GOSUB 5000:GOTO 260
1000 '----------- チョクホウタイ ノ ニュウリョク ----------------
1010 CLS
1020 INPUT "タテ ノ ナガサ = ",D
1030 IF D<=0 THEN 1010
1040 INPUT "ヨコ ノ ナガサ = ",W
1050 IF W<=0 THEN 1040
1060 INPUT "タカサ        = ",H
1070 IF H<=0 THEN 1060
1080 RETURN
1500 '----------- チョクホウタイ ヲ エガク ----------------
1510 PX=0:PY=H:PZ=D:GOSUB 3500
1520 LINE(XP,YP)-(XP,YP),PSET,PC
1530 PX=W:GOSUB 3500:LINE-(XP,YP)
1540 PZ=0:GOSUB 3500:LINE-(XP,YP)
1550 PX=0:GOSUB 3500:LINE-(XP,YP)
1560 PZ=D:GOSUB 3500:LINE-(XP,YP)
1570 PY=0:GOSUB 3500:LINE-(XP,YP)
1580 PX=W:GOSUB 3500:LINE-(XP,YP)
1590 PY=H:GOSUB 3500:LINE-(XP,YP)
```

```
1600 PY=0:GOSUB 3500:LINE(XP,YP)-(XP,YP),PSET,PC
1610 PZ=0:GOSUB 3500:LINE-(XP,YP)
1620 PY=H:GOSUB 3500:LINE-(XP,YP)
1630 RETURN
2000 '----------  ﾏｯﾋﾟﾝｸﾞ ﾙｰﾁﾝ  --------------------
2010 LOCATE 0,9
2020 INPUT"SCREEN X START= ",XS
2030 INPUT"SCREEN X END   = ",XE
2040 INPUT"SCREEN Y START= ",YS
2050 INPUT"SCREEN Y END   = ",YE
2060 IF XS<0 OR YS<0 THEN 2000
2070 IF XE>639 OR YE>199 THEN 2000
2080 IF XS>XE OR YS>YE THEN 2000
2090 PRINT:PRINT"MAPPING AREA"
2100 PRINT"  1. ｼｮｳﾒﾝ"
2110 PRINT"  2. ﾘﾂﾒﾝ "
2120 PRINT"  3. ｿｸﾒﾝ "
2130 PRINT"  4. END "
2140 A$=INKEY$(1)
2150 IF A$="1" OR A$="2" OR A$="3" THEN 2170
2160 IF A$="4" THEN RETURN ELSE 2140
2170 PRW &HFF:MS=VAL(A$)
2180 ON MS GOSUB 2200,2400,2600
2190 PRW 0:GOTO 2000
2200 '----------  ﾏｯﾋﾟﾝｸﾞ ﾙｰﾁﾝ (ｼｮｳﾒﾝ)  ----------
2210 PRW &HFF
2220 LX=XE-XS:LY=YE-YS:SCX=W/LX:SCY=H/LY:PZ=D
2222 IF SCY<1 THEN STP2=1 ELSE STP2=1/SCY
2224 IF SCX<1 THEN STP3=1 ELSE STP3=1/SCX
2230 FOR J=0 TO LY STEP STP2
2240   PY=H-J*SCY
2250   FOR I=0 TO LX STEP STP3
2260     SCREEN 0,2,0
2270     CC=POINT(INT(XS+I+.5),INT(YS+J+.5))
2280     SCREEN 0,0,0
2290     PX=I*SCX:GOSUB 3500
2300     PSET (XP,YP,CC):PSET (XP+1,YP,CC)
2310   NEXT
2320 NEXT
2330 RETURN
2400 '----------  ﾏｯﾋﾟﾝｸﾞ ﾙｰﾁﾝ (ﾘﾂﾒﾝ)  ----------
2410 PRW &HFF
2420 LX=XE-XY:LY=YE-YS:SCX=W/LX:SCY=D/LY:PY=H
2430 IF SCY<1 THEN STP2=1 ELSE STP2=1/SCY
2440 IF SCX<1 THEN STP3=1 ELSE STP3=1/SCX
2450 FOR J=0 TO LY STEP STP2
2460   PZ=J*SCY
2470   FOR I=0 TO LX STEP STP3
2480     SCREEN 0,2,0
2490     CC=POINT(INT(XS+I+.5),INT(YS+J+.5))
2500     SCREEN 0,0,0
2510     PX=I*SCX:GOSUB 3500
2520     PSET (XP,YP,CC):PSET (XP+1,YP,CC)
```

```
2530   NEXT
2540 NEXT
2550 RETURN
2600 '----------  ﾏｯﾋﾟﾝｸﾞ ﾙｰﾁﾝ (ｿｸﾒﾝ)  -----------
2610 PRW &HFF
2620 LX=XE-XS:LY=YE-YS:SCX=D/LX:SCY=H/LY:PX=W
2630 IF SCY<1 THEN STP2=1 ELSE STP2=1/SCY
2640 IF SCX<1 THEN STP3=1 ELSE STP3=1/SCX
2650 FOR J=0 TO LY STEP STP2
2660   PY=H-J*SCY
2670   FOR I=0 TO LX STEP STP3
2680     SCREEN 0,2,0
2690     CC=POINT(INT(XS+I+.5),INT(YS+J+.5))
2700     SCREEN 0,0,0
2710     PZ=D-I*SCX:GOSUB 3500
2720     PSET (XP,YP,CC):PSET (XP+1,YP,CC)
2730   NEXT
2740 NEXT
2750 RETURN
3000 '----------  3D ｼﾞｮｳｹﾝ ﾆｭｳﾘｮｸ  -------------
3010 INPUT"X -- MOVE  = ",MVX
3020 INPUT"Y -- MOVE  = ",MVY
3030 INPUT"X -- ANGLE = ",RTX
3040 INPUT"Y -- ANGLE = ",RTY
3050 INPUT"Z -- ANGLE = ",RTZ
3060 SS=SIN(RAD(RTX)):CS=COS(RAD(RTY))
3070 SF=SIN(RAD(RTY)):CF=COS(RAD(RTY))
3080 SP=SIN(RAD(RTZ)):CP=COS(RAD(RTZ))
3090 INPUT"   Color   = ",PC
3100 IF (PC<0)+(PC>7) THEN 3090
3110 RETURN
3500 '----------  3D ﾉ ｹｲｻﾝ  -------------------
3510 XP=PX*CF*CP+PY*(SS*SF*CP-CS*SP)
3520 XP=XP+PZ*(CS*SF*CP+SS*SP)
3530 YP=PX*CF*SP+PY*(SS*SF*SP+CS*CP)
3540 YP=YP+PZ*(CS*SF*SP-SS*CP)
3550 ZP=-PX*SF+PY*SS*CF+PZ*CS*CF
3560 XP=XO+XP*2+MVX
3570 YP=YO-YP-MVY
3580 RETURN
4000 '----------  ｽｸﾘｰﾝ ｶﾞﾒﾝ ﾉ ﾛｰﾄﾞ  -------------
4010 CLS:PRINT"[ SCREEN LOAD ]"
4020 INPUT"  FILE NAME ? ",T$
4030 IF LEN(T$)>16 THEN 4000
4040 OPTION SCREEN 4:PRW &HFF
4050 OPEN"I",1,T$:REC=0
4060 GOSUB 6510:IF WI<>80 THEN 4000
4070 A$=INPUT$(128,1):B$=INPUT$(128,1)
4080 DEVO$"MEM1:",REC,A$,B$:REC=REC+1
4090 IF REC<&HC0 THEN 4070
4100 CLOSE #1:PRW 0:INIT:OPTION SCREEN 0
4110 FOR I=0 TO 7
4120   PALET I,PAL(I)
```

```
4130 NEXT:RETURN
5000 '----------  ｶﾞﾒﾝ ﾉ ﾛｰﾄﾞ  ------------------
5010 CLS:PRINT"[ LOAD ]"
5020 INPUT"  FILE NAME ? ('/'=RETURN) ",T$
5030 IF LEN(T$)>16 THEN 5000
5040 IF T$="/" THEN RETURN
5050 GOSUB 5090
5060 FOR I=0 TO 7
5070   PALET I,PAL(I)
5080 NEXT:RETURN
5090 OPTION SCREEN 4:PRW &HFF
5100 OPEN"I",1,T$:REC=0
5110 GOSUB 6510:IF WI<>80 THEN RETURN 5000
5120 A$=INPUT$(128,1):B$=INPUT$(128,1)
5130 DEVO$"MEM:",REC,A$,B$:REC=REC+1
5140 IF REC<&HC0 THEN 5120
5150 CLOSE #1:PRW 0:INIT
5160 OPTION SCREEN 0:CLS:RETURN
6000 '----------  ｶﾞﾒﾝ ﾉ ｾｰﾌﾞ  ------------------
6010 PRW 0:CLS
6020 PRINT"[ SAVE ]"
6030 INPUT"  FILE NAME ? ('/'=RETURN) ",T$
6040 IF LEN(T$)>16 THEN 6000
6050 IF T$="/" THEN RETURN
6060 OPTION SCREEN 4:PRW &HFF
6070 OPEN"O",1,T$:REC=0
6080 PRINT #1,80
6090 FOR I=0 TO 7
6100   PRINT #1,PAL(I)
6110 NEXT
6120 DEVI$"MEM:",REC,A$,B$:PRINT#1,A$;B$;
6130 REC=REC+1
6140 IF REC<&HC0 THEN 6120
6150 GOTO 5150
6500 '----------  WIDTH ﾁｪｯｸ  ------------------
6510 INPUT #1,WI
6520 IF WI<>80 THEN 6560
6530 FOR I=0 TO 7
6540   INPUT #1,PAL(I)
6550 NEXT:RETURN
6560 GOSUB 5150
6570 PRINT"WIDTH IS NOT 80 !! (Push Any key) ";
6580 KY$=INKEY$(1)
6590 RETURN
```

X1シリーズ用変更リスト

X1シリーズを使う場合は右の各行を変更します。

```
2070 IF XE>319 OR YE>199 THEN 2000

2260      SCREEN 0,1,0
2300      PSET(XP,YP,CC)

2480      SCREEN 0,1,0
2520      PSET(XP,YP,CC)

2680      SCREEN 0,1,0
2720      PSET(XP,YP,CC)
```

回転体へのマッピング・プログラム

回転体へのマッピング処理は，先の４例と同様にテクスチャパターン画面上に描かれたピクセル情報を，回転体のそれぞれの座標値へと変換させます。

回転体は，半径のＸ座標と高さのＹ座標を，それぞれ $R_X(\ )$，$R_Y(\ )$ に代入し，同時に長さを計算し，$RL(\ )$ に代入し，形状記述します。

回転体へのマッピング処理は，テクスチャパターン画面上に描かれたテクスチャから必要なパターンのＸ方向の長さを L_X，Ｙ方向の長さを L_Y，回転体の半径を $R_X(K)$，Ｙ座標を $R_Y(K)$（ただし $R_Y(K)$ は，最初の点からトータルな長さを表します），また，最初から最後までの長さを，

$L(=RL(N-1))$　　　　ただしＮは個数

と表すと，

$$\begin{cases} SC_X=\pi R_1/L_X \\ SC_Y=L/L_Y \\ LL=L-J\cdot SC_Y \end{cases}$$

と書き表すことができます。

LL がKと $K-1$ 番目の間にあるKを求めると，

$$P_Y=P_Y(K-1)+(R_Y(K)-R_Y(K-1))(LL-RL(K-1)/(RL(K)-RL(K-1))$$

半径 R_1 は，

$$R_1=R_X(K-1)+(R_Y(K)-R_X(K-1)(LL-RL(K-1))/(RL(K)-RL(K-1))$$

また，

$$P_X=-R_1\cos\theta$$

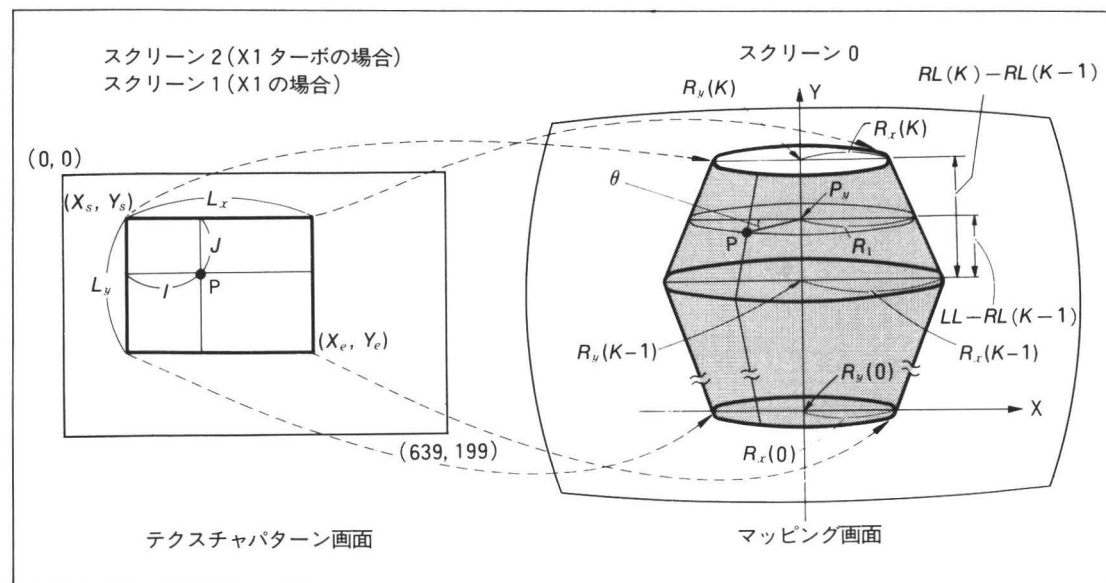



図４—22　回転体へのマッピング

$P_Z = R_1 \sin\theta$

ただし $\theta = I / LX \cdot \pi$（ラジアン）

と表すことができます。

プログラムの操作方法

このプログラムを入力して実行（RUN）させると，画面上に，

```
〔SCREEN  LOAD〕
FILE  NAME  ?
```

と聞いてくるので，他のマッピング・プログラムと同様に，画像データのうちマッピングしたい画像のファイル名を入力してください。読み込みが完了すると，マッピング処理のための物体形状の記述に入ります。

画面上には，

```
No.  =
タカサ   =
ハンケイ  =
```

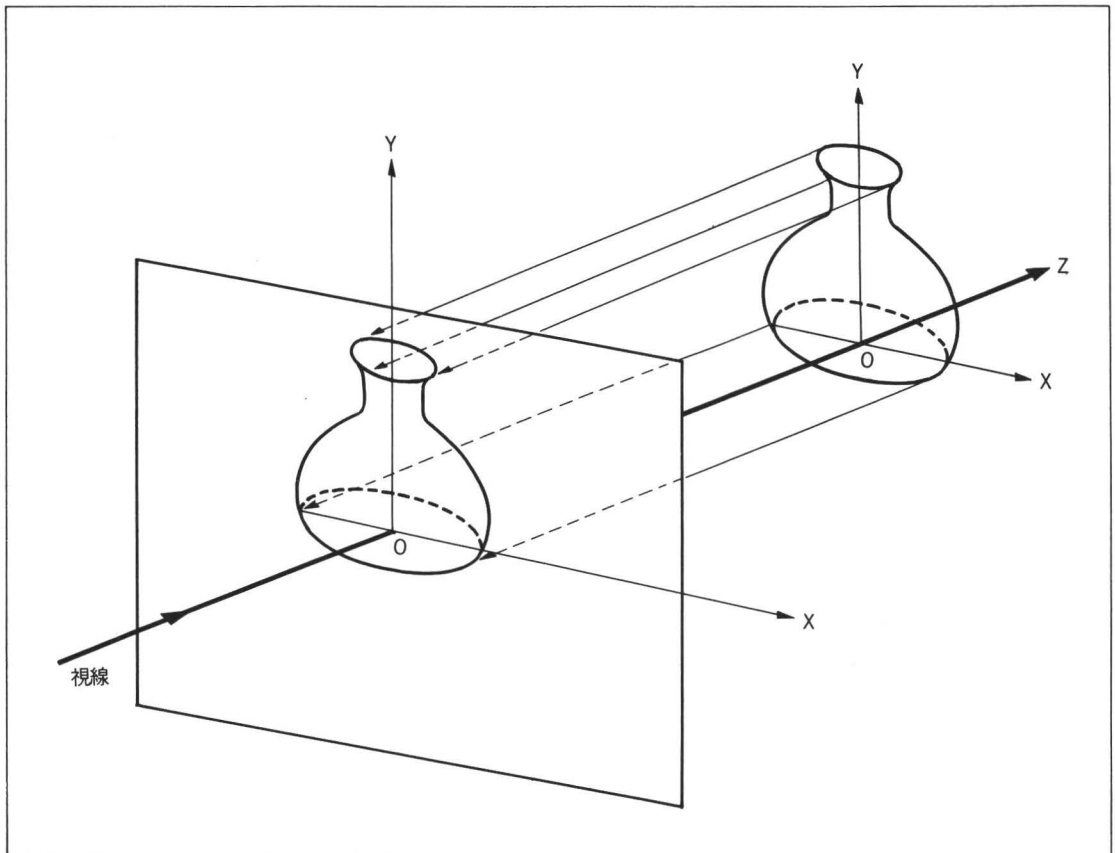



図4—23 ワールド座標／スクリーン座標と視線の関係

と聞いてくるので，Y軸を中心とした回転する点の座標位置を，点の番号（N），原点からの高さ（H$），原点からの距離＝半径（R）を順次入力してください。回転体の点は，100個まで入力が可能です。入力が終了すると，[E]キーを入力してください。

次いで，スクリーン座標のどこに描くのか決定します。前述のマッピングルーチンと同様に，コマンド指示に従って，次の値を入力してください。

```
X——MOVE   =
Y——MOVE   =
X——ANGLE  =
Y——ANGLE  =
Z——ANGLE  =
Color     =
```

次いで，

```
SCREEN  X  START  =
SCREEN  X  END    =
SCREEN  Y  START  =
SCREEN  Y  END    =
```

と聞いてくるので，テクスチャパターン画面のマッピング表示領域を決めてください（指示領域は，前述のマッピングルーチンと同様です）。指示に従って，マッピング処理が回転体の上面から開始されます。

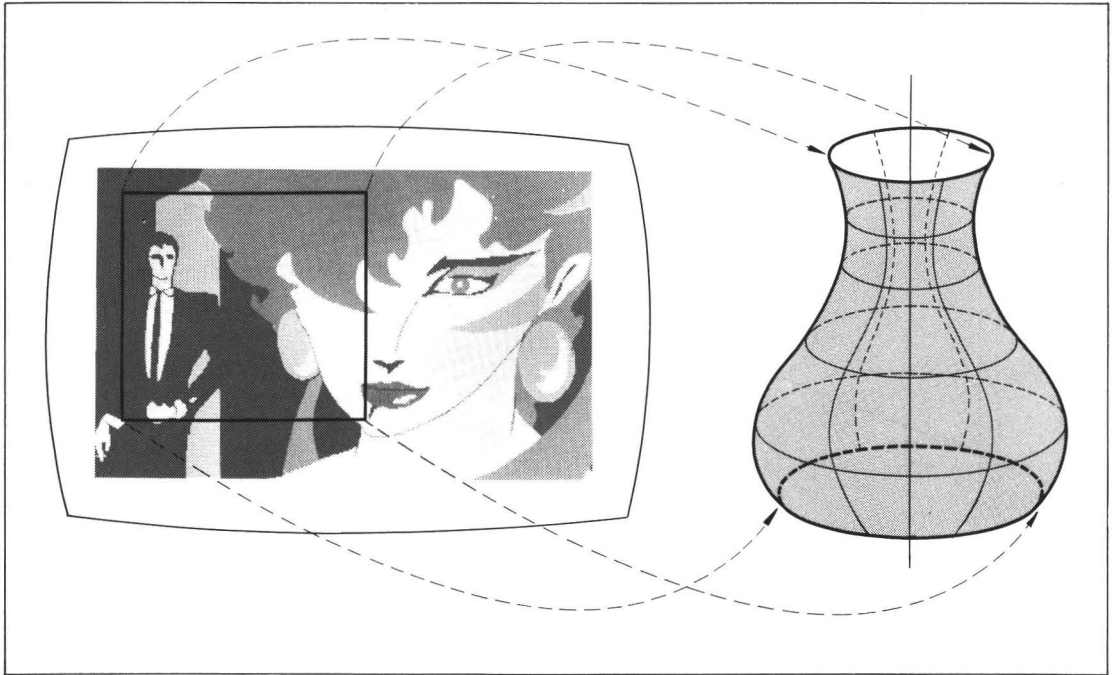



図4—24　回転体へのマッピング処理

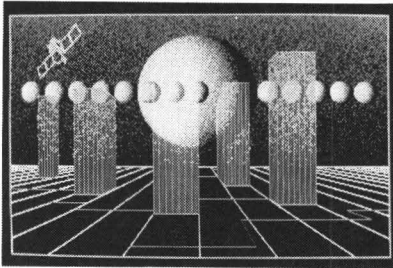
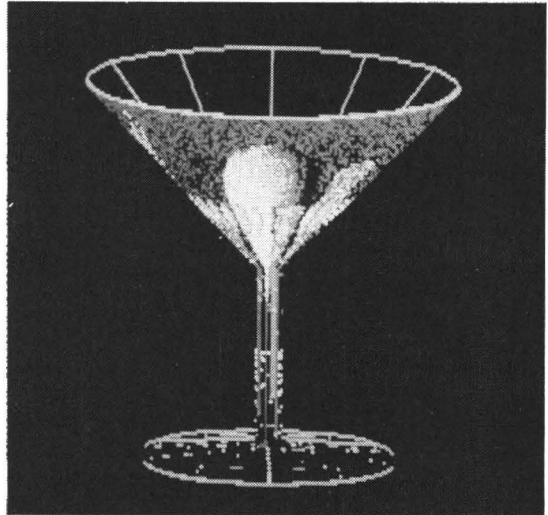
マッピングが完了すると，

```
〔SAVE〕
FILE  NAME  ?  ('/'=RETURN)
```

と聞いてくるので，16文字以内でファイル名を付けてSAVEしてください。ファイル・ディスクリプタを指定すると，それぞれのデバイスにファイルが可能です。

プログラムの内容

プログラムの組み立ては，116ページの円柱の場合とほぼ同じなので，参照してください。

回転体へのマッピング 出力例

テクスチャパターン

テクスチャパターン

変数とその内容			
PAL（ ）	パレットコード	XS	マッピング用画面の左端
XO	スクリーン中央の位置（X座標）	YS	マッピング用画面の上端
YO	スクリーン中央の位置（Y座標）	XE	マッピング用画面の右端
STP	円弧の角度（STEP）	YE	マッピング用画面の下端
STP1		CC	マッピング位置のカラー
PC	カラーコード	MVX	スクリーンX方向の移動
PX	物体の位置（X座標）	MVY	スクリーンY方向の移動
PY	物体の位置（Y座標）	RTX	物体のX軸に対する回転
PZ	物体の位置（Z座標）	RTY	物体のY軸に対する回転
XP	3D変換後のスクリーン（X座標）	RTZ	物体のZ軸に対する回転
YP	3D変換後のスクリーン（Y座標）		

回転体マッピング・プログラムリスト

```
1 '*************************************
2 '*                                   *
3 '*      MAPPING - ROTATION      gak   *
4 '*                                   *
5 '*       Programmed By [ SAIMU ]      *
6 '*                                   *
7 '*           1985 . 4 . 1             *
8 '*                                   *
9 '*************************************
100 '----------- ｶﾞﾒﾝ ｾｯﾃｲ ----------------------
110 KLIST 0:OPTION SCREEN 0:WIDTH 80,25,0,0:CLS
120 DIM PAL(7)
130 XO=320:YO=100:STP=10:STP1=30
140 FOR I=0 TO 7:PAL(I)=I:NEXT
200 '----------- ﾒｲﾝ ﾙｰﾁﾝ ----------------------
210 SCREEN 2,2,0:GOSUB 4000
220 SCREEN 0,0,0:GOSUB 1000:GOSUB 3000
230 IF PC=0 THEN 250
240 COLOR PC:GOSUB 1500
250 COLOR 7 :GOSUB 2000
260 PRW 0:CLS:PRINT "[MAPPING FILE ]"
270 PRINT " 1. SCREEN SAVE"
280 PRINT " 2. SCREEN LOAD"
290 PRINT " 3. END"
300 PRINT "   Push Key 1 OR 2 OR 3 ";
310 A$=INKEY$(1)
320 ON INSTR("123",A$) GOSUB 360,370,340
330 GOTO 310
340 INIT:END
350 '--------------------
360 GOSUB 6000:GOTO 260
370 GOSUB 5000:GOTO 260
1000 '----------- ｶｲﾃﾝﾀｲ ﾉ ﾆｭｳﾘｮｸ ---------------
1010 CLS:DIM RL(100),RX(100),RY(100)
1020 N=1:RL(0)=0
1030 PRINT "No.";N;
1040 INPUT " ﾀｶｻ =",H$
1050 IF H$="E" THEN 1150
1060 IF VAL(H$)>100 OR VAL(H$)<-99 THEN 1030
1070 INPUT " ﾊﾝｹｲ=",R
1080 IF R<=0 THEN 1070
1090 RX(N-1)=R:RY(N-1)=VAL(H$)
1100 IF N=1 THEN 1130
1110 L=SQR((RX(N-1)-RX(N-2))^2+(RY(N-1)-RY(N-2))^2)
1120 RL(N-1)=RL(N-2)+L
1130 N=N+1:IF N>100 THEN 1150
1140 GOTO 1030
```

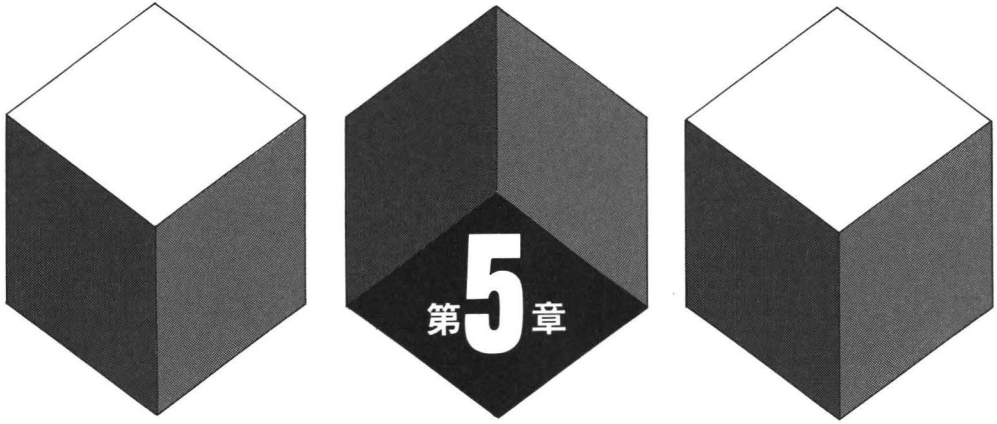
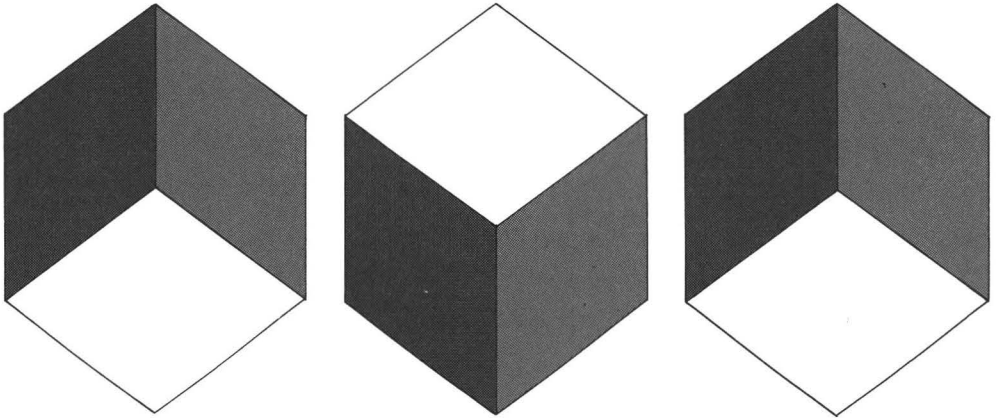
```
1150 N=N-1:IF N=1 THEN 1020
1160 L=RL(N-1)
1170 RETURN
1500 '---------- ｶｲﾃﾝﾀｲ ｦ ｴｶﾞｸ ----------------
1510 PRW &HFF
1520 FOR Z=0 TO N-1
1530   PY=RY(Z):PX=RX(Z):PZ=0:GOSUB 3500
1540   LINE (XP,YP)-(XP,YP),PSET,PC
1550   FOR TH=0 TO 360 STEP STP
1560     PX=RX(Z)*COS(RAD(TH))
1570     PZ=RX(Z)*SIN(RAD(TH))
1580     GOSUB 3500:LINE -(XP,YP)
1590   NEXT
1600 NEXT
1610 FOR TH=0 TO 360 STEP STP1
1620   PX=RX(0)*COS(RAD(TH)):PY=RY(0)
1630   PZ=RX(0)*SIN(RAD(TH)):GOSUB 3500
1640   LINE (XP,YP)-(XP,YP),PSET,PC
1650   FOR Z=0 TO N-1
1660     PX=RX(Z)*COS(RAD(TH)):PY=RY(Z)
1670     PZ=RX(Z)*SIN(RAD(TH))
1680     GOSUB 3500:LINE -(XP,YP)
1690   NEXT
1700 NEXT
1710 PRW 0:RETURN
2000 '---------- ﾏｯﾋﾟﾝｸﾞ ﾙｰﾁﾝ ----------------
2010 CLS
2020 INPUT"SCREEN X START= ",XS
2030 INPUT"SCREEN X END   = ",XE
2040 INPUT"SCREEN Y START= ",YS
2050 INPUT"SCREEN Y END   = ",YE
2060 IF XS<0 OR YS<0 THEN 2000
2070 IF XE>639 OR YE>199 THEN 2000
2080 IF XS>XE OR YS>YE THEN 2000
2090 '---------- ﾏｯﾋﾟﾝｸﾞ ｶｲﾃﾝﾀｲ ----------------
2100 PRW &HFF:LX=XE-XS:LY=YE-YS
2110 SCY=L/LY:K=1
2120 IF SCY<1 THEN STP2=1 ELSE STP2=1/SCY
2130 FOR J=LY TO 0 STEP -STP2
2140   LL=L-J*SCY
2150   IF LL>RL(K) THEN K=K+1:IF LL>RL(K) THEN 2140
2160   PY=(RY(K)-RY(K-1))*(LL-RL(K-1))
2170   PY=RY(K-1)+PY/(RL(K)-RL(K-1))
2180   R1=(RX(K)-RX(K-1))*(LL-RL(K-1))
2190   R1=RX(K-1)+R1/(RL(K)-RL(K-1))
2200   SCX=PAI(R1)/LX
2210   IF SCX<1 THEN STP3=1 ELSE STP3=1/SCX
2220   FOR I=0 TO LX STEP STP3
2230     SCREEN 0,2,0
```

```
2240      CC=POINT(INT(XS+I+.5),INT(YS+J+.5))
2250      SCREEN 0,0,0
2260      PX=-R1*COS(PAI(I/LX)):PZ=R1*SIN(PAI(I/LX))
2270      GOSUB 3500
2280      PSET (XP,YP,CC):PSET (XP+1,YP,CC)
2290   NEXT
2300 NEXT
2310 PRW 0:RETURN
3000 '----------- 3D ｼﾞｮｳｹﾝ ﾆｭｳﾘｮｸ ---------------
3010 INPUT"X -- MOVE  = ",MVX
3020 INPUT"Y -- MOVE  = ",MVY
3030 INPUT"X -- ANGLE = ",RTX
3040 INPUT"Y -- ANGLE = ",RTY
3050 INPUT"Z -- ANGLE = ",RTZ
3060 SS=SIN(RAD(RTX)):CS=COS(RAD(RTY))
3070 SF=SIN(RAD(RTY)):CF=COS(RAD(RTY))
3080 SP=SIN(RAD(RTZ)):CP=COS(RAD(RTZ))
3090 INPUT"   Color   = ",PC
3100 IF (PC<0)+(PC>7) THEN 3090
3110 RETURN
3500 '----------- 3D ﾉ ｹｲｻﾝ -----------------------
3510 XP=PX*CF*CP+PY*(SS*SF*CP-CS*SP)
3520 XP=XP+PZ*(CS*SF*CP+SS*SP)
3530 YP=PX*CF*SP+PY*(SS*SF*SP+CS*CP)
3540 YP=YP+PZ*(CS*SF*SP-SS*CP)
3550 ZP=-PX*SF+PY*SS*CF+PZ*CS*CF
3560 XP=XO+XP*2+MVX
3570 YP=YO-YP-MVY
3580 RETURN
4000 '----------- ｽｸﾘｰﾝ ｶﾞﾒﾝ ﾉ ﾛｰﾄﾞ ---------------
4010 CLS:PRINT"[ SCREEN LOAD ]"
4020 INPUT"  FILE NAME ? ",T$
4030 IF LEN(T$)>16 THEN 4000
4040 OPTION SCREEN 4:PRW &HFF
4050 OPEN"I",1,T$:REC=0
4060 GOSUB 6500:IF WI<>80 THEN 4000
4070 A$=INPUT$(128,1):B$=INPUT$(128,1)
4080 DEVO$"MEM1:",REC,A$,B$:REC=REC+1
4090 IF REC<&HC0 THEN 4070
4100 CLOSE #1:PRW 0:INIT:OPTION SCREEN 0
4110 FOR I=0 TO 7
4120   PALET I,PAL(I)
4130 NEXT:RETURN
5000 '----------- ｶﾞﾒﾝ ﾉ ﾛｰﾄﾞ ---------------------
5010 CLS:PRINT"[ LOAD ]"
5020 INPUT"  FILE NAME ? ('/'=RETURN) ",T$
5030 IF LEN(T$)>16 THEN 5000
5040 IF T$="/" THEN RETURN
5050 GOSUB 5090
```

```
5060 FOR I=0 TO 7
5070   PALET I,PAL(I)
5080 NEXT:RETURN
5090 OPTION SCREEN 4:PRW &HFF
5100 OPEN"I",1,T$:REC=0
5110 GOSUB 6500:IF WI<>80 THEN RETURN 5000
5120 A$=INPUT$(128,1):B$=INPUT$(128,1)
5130 DEVO$"MEM:",REC,A$,B$:REC=REC+1
5140 IF REC<&HC0 THEN 5120
5150 CLOSE #1:PRW 0:INIT
5160 OPTION SCREEN 0:CLS:RETURN
6000 '---------- ｶﾞﾒﾝ ﾉ ｾｰﾌﾞ ----------------------
6010 PRW 0:CLS
6020 PRINT"[ SAVE ]"
6030 INPUT"  FILE NAME ? ('/'=RETURN) ",T$
6040 IF LEN(T$)>16 THEN 6000
6050 IF T$="/" THEN RETURN
6060 OPTION SCREEN 4:PRW &HFF
6070 OPEN"O",1,T$:REC=0
6080 PRINT #1,80
6090 FOR I=0 TO 7
6100   PRINT #1,PAL(I)
6110 NEXT
6120 DEVI$"MEM:",REC,A$,B$:PRINT#1,A$;B$;
6130 REC=REC+1
6140 IF REC<&HC0 THEN 6120
6150 GOTO 5150
6490 '---------- WIDTH ﾁｪｯｸ ----------------------
6500 INPUT #1,WI
6510 IF WI<>80 THEN 6550
6520 FOR I=0 TO 7
6530   INPUT #1,PAL(I)
6540 NEXT:RETURN
6550 GOSUB 5150
6560 PRINT"WIDTH IS NOT 80 !! (Push Any key) ";
6570 KY$=INKEY$(1)
6580 RETURN
```

X1シリーズ用変更リスト | X1シリーズを使う場合は次の各行を変更します。

```
2070 IF XE>319 OR YE>199 THEN 2000

2230     SCREEN 0,1,0
2280     PSET (XP,YP,CC)
```

図形処理技術の応用

レイトレーシング

私たちのまわりには，限りない光が存在しています。光の本質はどうなのかということは，従来より光学の分野で研究されてきています。それについて解説することは，この本の目的ではないと思われますので省きますが，とりもなおさず光によって，私たちは目の前にある物体の存在を知ることができるわけです。

コンピュータに物体を形成させるうえでも，よりリアルな（真実味のある）モデルを作成するためには，光の様子をどのようにして計算させればよいかが大きな課題となります。光源からの光が物体に当たった場合に，どのように見えるのか，どのように映るのかが，数学的に計算される必要があるわけですが，これは従来より，幾何光学の範囲で解明されているので，それを参考にしてプログラム化していきます。

幾何光学では，光を1本の直進する線としてとらえていますが，その光線の当たり具合によって物体の棒組がどう変化するかをシミュレートしてみるという考え方をとります。1本の光線は，直進性があり，物体に当たった場合にはその材質に応じて反射したり，透過したり，屈折したりします。これらの現象には，すでに数学的にモデル化されたものが存在しており，そういった手法で物体表面上の輝度を計算することによって，コンピュータ内部にデータを構築するわけです。

光源からの光が突き進んでいくうちに，物体に当たるか当たらないかを計算し，当たればそれが鏡のようなもので全反射するのか，吸収されてしまうのか，透き通るのか……といった光の軌跡を追って，それによってどのように形態が作られているのかシミュレートする方法を，光線追跡法といいます。つまりレイトレーシングと呼んでいます。レイトレーシングのアルゴリズムには，さまざまな方法があります。

ここでは，まず3次元グラフィックスの応用として，基礎的な事がらを学んでいただこうという意図で，不透明な球体に光が当たった場合にどのように変化するかということについて解説し，そのアルゴリズムを紹介します。光線の当たる角度に応じて物体がどう変化するか，光線の輝度の強さによってどのように変わっていくのかシミュレートしてみましょう。

なお，本書では，3次元グラフィックスの入門書としての位置づけをしているため，レイトレーシングのその他のテクニック，例えば物体の材質が半透明だったり，透明だったりしたらどうなるか，影の床面への映り込み（シェイディング）はどうするのか，テクスチャマッピングと組み合わせた場合はどうするのか，その他の自由な曲面に対してはどうするのか等については，他書に譲ります。

レイトレーシングによる画像の作り方

レイトレーシングは，アメリカのベル研究所のターナー・ホイッティッドによって開発されたもので，表示される物体全面の光の当たり具合の情報を，各画素における明度として計算するものです。1つのスクリーン上の画素の明るさを計算するために，視点から光源までを逆にたどっていき，視点からのその画素を通る視線が出合う物体の面の情報と，そこから反射して至る次の面，そしてその次

の面と最終的に至達する光源の情報といったものによって決定されます。

これは，人間の側からの視覚としては，自然に理解しやすい考え方であり，理論としても非常に単純ですが，すべての画素について視線の計算を行うために，計算時間が非常に長くかかるという欠点もあります。しかし，あらゆる物体についてのリアルなシミュレーションが可能なため，非常に大事な手法ということができます。

本格的な方法としては，図5—1のように考えると，第1章39～40ページに示した，次の2つの式が導かれます。

$$I = I_a + \sum_l I_r l + I_t$$
$$I_r l = \{kd \cos i + ks(\cos j)^n\} \cdot I_l$$

これは，先ほどのターナー・ホイッティッドによって実証されています。彼が実際に，透明なガラス球と光沢のある球体を市松模様の上に浮かび上がらせ，それに上方からの光によるハイライトや，床面への影，透明球から透かして見える床面のゆがみ，光沢のある球体に映る床面の模様といったシミュレーションをみごとに表現して上の数式を実証しています。これには，数々のテクニックがいるわけですが，本書ではその初歩のテクニックとして，光源からの光が球体にどう反映されるかについてのみ解説しますので，方程式としては簡単な構造となります。

通常，光には電球などの点光源から出る光と，螢光灯等の線光源からくる光，それに，太陽光のように平行光線からくる光とが存在します。ここでは，解説をわかりやすくするために，平行光線についてのみ考えます。平行光線は，場所によらず一定の方向を向いている光線であるといえます。平行光線は，場所に関係なく一定の方向を向いた光線であり，その各成分をベクトル$\overrightarrow{\mathrm{L}}$，球体の法線をベクトル$\overrightarrow{\mathrm{N}}$とすると，表面の輝度$I$は，次の関係式で表されます（図5—2)。

$$I = I \propto (\overrightarrow{L} \cdot \overrightarrow{N})$$

これは，ランバーの法則（Lamber's Law）と呼ばれているものです。ここで比例定数は，照明の

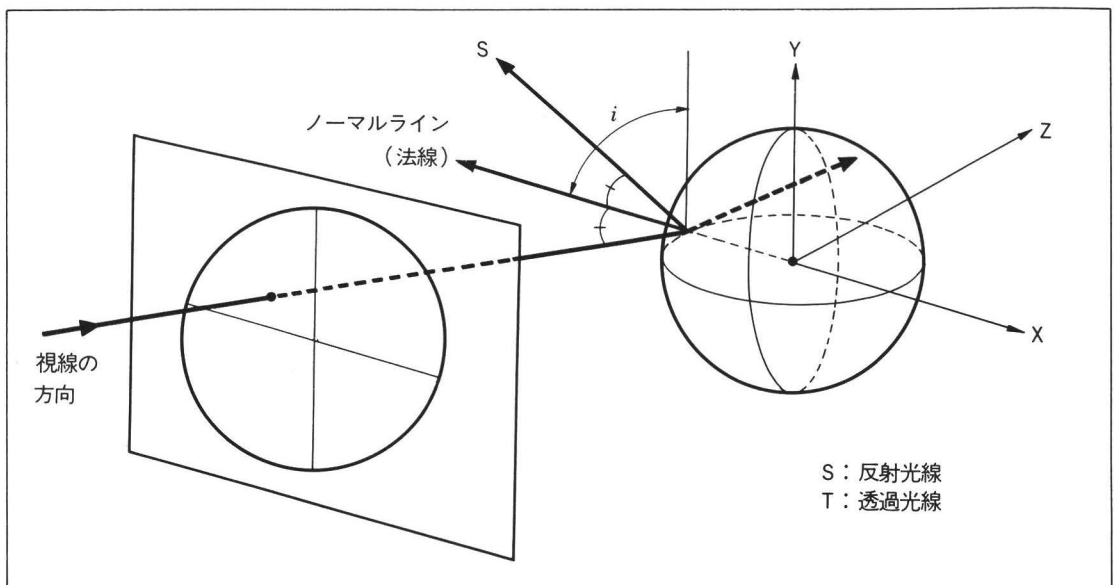



図5—1　レイトレーシングの考え方

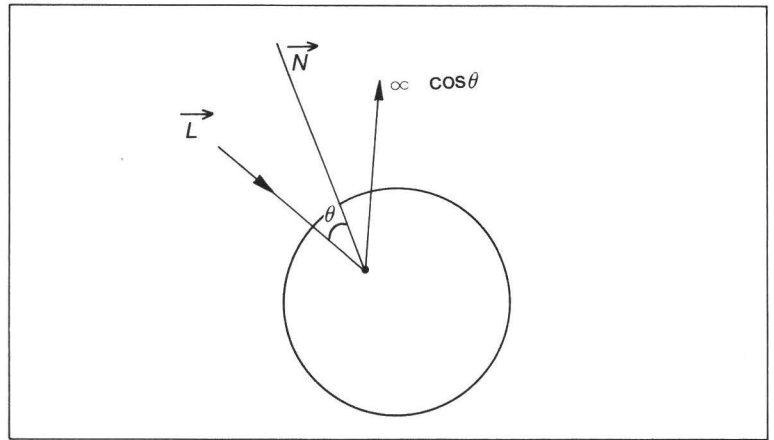



図5—2　ランバーの法則

強さ，乱反射により規定します。ここでは，この式によって，球面上の輝度を計算していきます。

ここで各球面の色が問題になってきます。幸か不幸か，X１ターボおよびX１シリーズのマイコンには８色の色数しか存在していません（ほとんどのマイコンがそうですが）。これは，図5—3のようにR（赤），G（緑），B（青）各色に対して１ビット（０か１かの２値）しか持ち合わせておらず，これらの組み合わせによって８色の表示をしているわけです。こういったパソコンで多くの色を表現するためには，ディザ法（dithering）というテクニックを行います。

これは，一種のモザイクのようなもので，８色の色をうまく並べることによって必要な色を表現しようというものです。基本的には，規則正しい方法（組織的ディザ法）と，乱数を使用した不規則な方法（ランダム・ディザ法）に分かれます。

組織的ディザ法は，タイルペイントとして，すでに別著「パソコングラフィックスの作り方楽しみ方」（学研版）でも紹介済みです。これを使用すると，規則性があるため，普通の２次元のペイントの場合には効果を発揮するのですが，本書で扱うような３次元プリミティブを塗る場合には，かえってその周期的なパターンが，「くせ」となって不自然なものになってしまいます。そのため，ここでの中間色表示には，ランダム・ディザ法を使用します。

ランダム・ディザ法では，プリミティブにあたる各ピクセルの輝度に，スレッショルド値を乱数によって変化させて確率的に対応させます。例えば，X１の場合は，640×200ドットで画面が構成されています。これは縦横比の分割でいくと２倍の関係になっているので，横方向の２ドットを１画素

	G	R	B
0（黒色）	0	0	0
1（青色）	0	0	1
2（赤色）	0	1	0
3（マゼンタ）	0	1	1
4（緑色）	1	0	0
5（シアン）	1	0	1
6（黄色）	1	1	0
7（白色）	1	1	1

R
M
Y
W
G
B
C

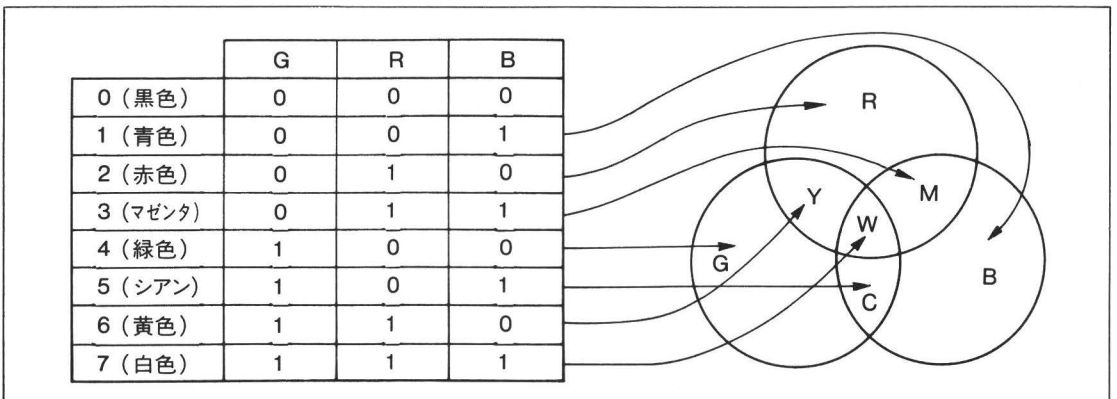



図5—3　X１ターボ／X１シリーズのカラー表示

として考えると，１画素に対し３段階の表現が可能になります。

この各画素にランダム・ディザ法を適用します。各画素の輝度は，スレッショルド値を変化させるために，組み込み関数によって，

$$-\frac{1}{2}N \sim +\frac{1}{2}N$$

の範囲で一様な乱数を作り出します。

通常，連続的にグレイスケールを得るのに必要なスレッショルドは，

$$TH1=B-\frac{1}{2}N \qquad TH2=B+\frac{1}{2}N$$

（ただし，Ｂはグレイスケールの中心の輝度（この場合は128），Ｎはコントラスト）

で表わされます。

これを使うことで中間色表示が滑らかな表示となります。

プログラムの操作方法

このプログラムを入力して実行（RUN）させると，画面上に，

```
バック　ノ　イロハ(0−7)　=
```

と聞いてくるので，まず画面の背景色を入力してください。

次いで，

```
キュウ　ノ　ハンケイ　=
```

と聞いてきますので，実際に描かせる球の半径を入力してください。さらに，

```
X—DIRECTION　MOVE　=
Y—DIRECTION　MOVE　=
```

と続けて聞いてくるので，描きたい球のスクリーン上の位置を入力してください。図５—４のようにスクリーン上の原点（０，０）の位置は，画面中央に設置，Ｘ正方向で中央より右方に，Ｙ正方向で中央より上方に球の中心を移動させます。例えば，Ｘ＝50，Ｙ＝−20を入力すると，スクリーン上

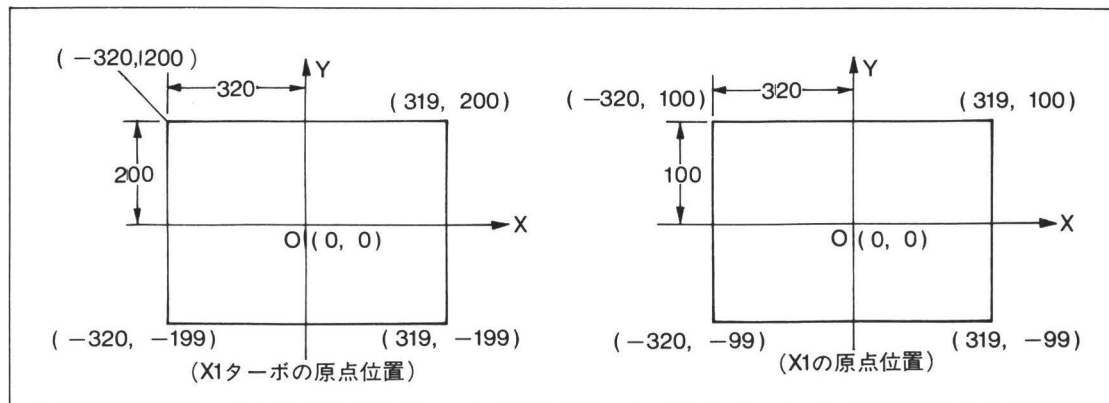



図５—４　レイトレーシング・プログラムのスクリーン座標系のＸ，Ｙ座標

では中央より右横方向に50，下方向に20移動した所を中心とした球を描くこととなります。

次に，

```
LIGHT  ANGLE(Y—AXIS)  =
```

と聞いてくるので，光線が球体に当たる角度を指定してください。スクリーン座標系で見た場合に，Y軸の光線角度が与えられます。例えば，0度を入力するために，[0][CR]とキー入力すると，光源は上方から下方に向けて光が当たり，ハイライト点は，図5—5の左の図のように真上に位置することになります。

次いで，

```
LIGHT  ANGLE(X—Z  PLANE)  =
```

と聞いてくるので，スクリーン座標系でのX軸の角度を入力してください。例えば，先ほどのLIGHT ANGLE（Y—AXIS）を90度とした場合は，図5-6のように角度指定によってハイライト点が変化します。

以上のように，ハイライト点の指示は，LIGHT ANGLEの2つの値の指示によって決めることができます。

次に，描く球体の色を指定するわけですが，BLUE（青色），RED（赤色），GREEN（緑色）それぞれを，256階調で分割し，合計1,677万7,216通りの組み合わせによって色表現を行います。ただX1ターボおよびX1は，8色の色データしか持っていませんので，ドットごとの混色によってカラーの組み合わせを行います。組み合わせによる色の濃度は各色0～255階調の混合比で入力します。現実の色では何色まで認識できるか試してください。

```
キュウタイ  ノ  イロ BLUE  (0－255)  =
キュウタイ  ノ  イロ RED(0－255)  =
キュウタイ  ノ  イロ GREEN(0－255)  =
```

各基本色の色データを混ぜあわせることにって，色指定ができたと思いますが，次いで，ハイライト点の輝度を指定することができます。

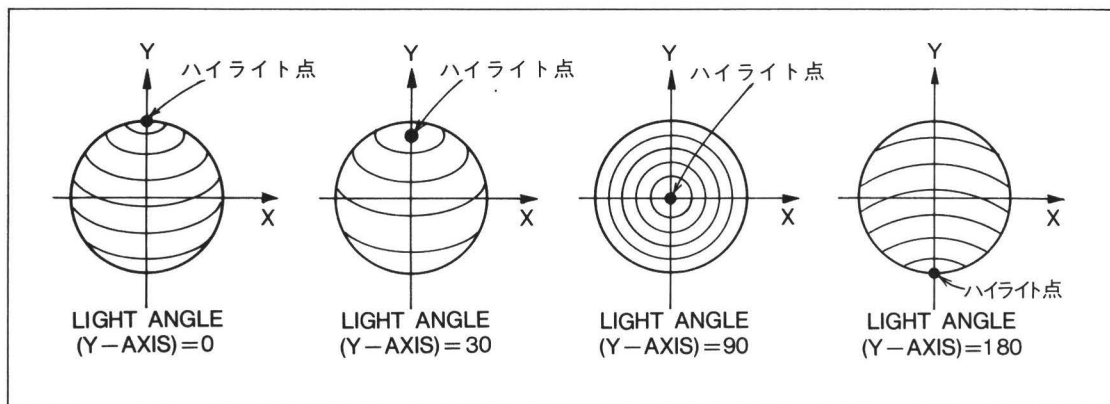



図5—5　LIGHT　ANGLE(Y-AXIS) による変換

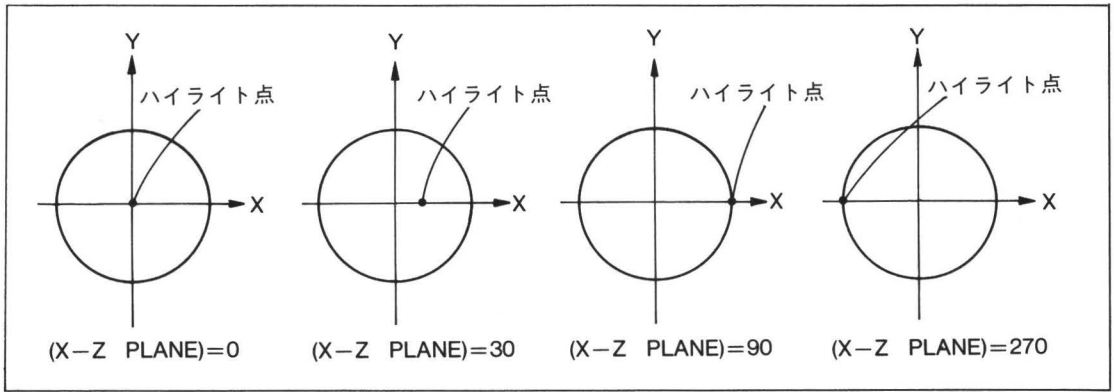
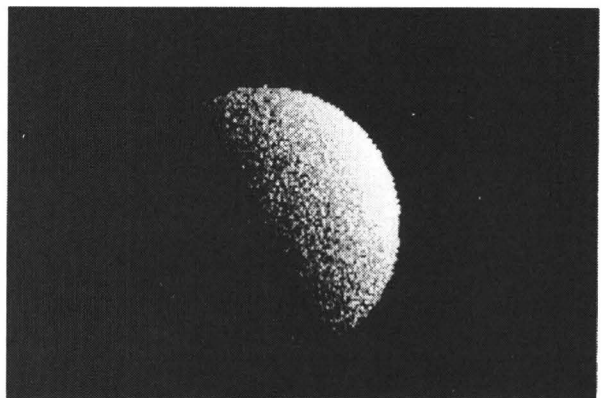
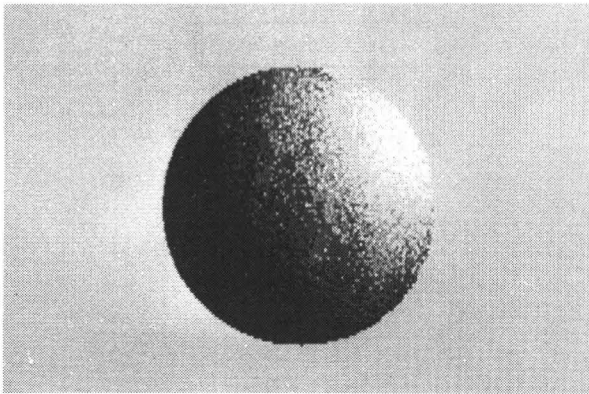
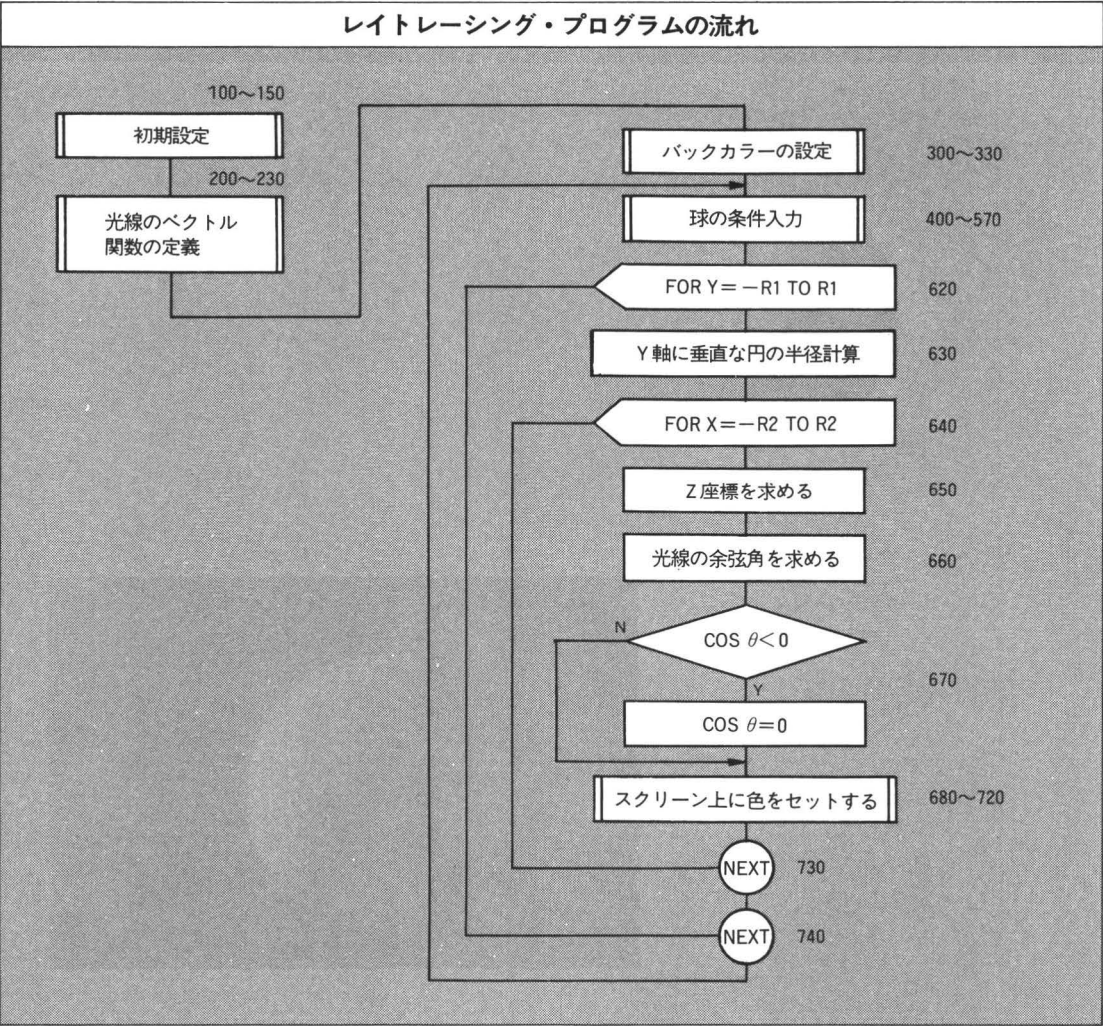



図5−6　LIGHT　ANGLE(X-Z　PLANE)による変換(Y-AXISが90度の場合の例)

ハイライト　キョウド(1−255)　=

と聞いてくるので，光を当てない場合は[0][CR]，順次光を強くしていき，最高に強く当てる場合は，[2][5][5][CR]と入力してください。256通りの階調で明度が与えられます。

レイトレーシング 出力例

変数とその内容			
WD	X1ターボとX1の係数	BETA	光線のX－Z面に対する角度
XO	スクリーン中央の位置(X座標)	BL	球の青色成分
YO	スクリーン中央の位置(Y座標)	RD	球の赤色成分
M	画像のノイズの振幅	GR	球の緑色成分
N	画像グレースケールのスレッショルド値	HL	反射光の輝度
N1	画像グレースケールのスレッショルド値	LX	光線のX方向ベクトル
C	背景色	LY	光線のY方向ベクトル
R1	球の半径	LZ	光線のZ方向ベクトル
R2	Y軸に垂直な面の円の半径	CS	光線の余弦
MVX	X方向の移動量	CL	球の各点の色
MVY	Y方向の移動量	SX	画面のX座標
ALPHA	光線のY軸に対する角度	SY	画面のY座標

レイトレーシング・プログラムの流れ

レイトレーシング・プログラムリスト

```
1 '************************************
2 '*                                  *
3 '*   RAY TRACEING for       v.1.0   *
4 '*     file name is [ RAY TRACE ]   *
5 '*                                  *
6 '*           1985. 4. 15            *
7 '*      Programed by [ SAIMU ]      *
8 '*                                  *
9 '************************************
100 '----------- ｼｮｷｾｯﾃｲ -------------------------
110 OPTIONSCREEN 0:KLIST 0                :'(X1 turbo)
120 WIDTH 80,25,1,2:INIT                  :'(X1 turbo)
130 'WIDTH 80:INIT                        :'(X1      )
140 WD=2:XO=320:YO=200                    :'(X1 turbo)
150 'WD=1:XO=320:YO=100                   :'(X1      )
160 M=128:N=64:N1=192
200 '----------- ｶﾝｽｳ ﾉ ﾃｲｷﾞ --------------------
210 DEF FNX(A,B)=SIN(RAD(A))*SIN(RAD(B))
220 DEF FNY(A)=COS(RAD(A))
230 DEF FNZ(A,B)=SIN(RAD(A))*COS(RAD(B))
300 '----------- ﾊﾞｯｸ ﾉ ｶﾗｰ ----------------------
310 INPUT "ﾊﾞｯｸ ﾉ ｲﾛ ﾊ (0-7) = ",C
320 IF C<0 OR C>7 THEN 310
330 LINE (0,0)-(639,WD*200-1),PSET,C,BF
400 '----------- ﾚｲ ﾄﾚｰｼﾝｸﾞ ﾆｭｳﾘｮｸ --------------
410 CLS:INPUT "ｷｭｳ ﾉ ﾊﾝｹｲ                = ",R1
420 IF R1<=0 THEN 410 ELSE R1=R1/WD
430 INPUT "X-DIRECTION MOVE        = ",MVX
440 INPUT "Y-DIRECTION MOVE        = ",MVY
450 INPUT "LIGHT ANGLE (Y-AXIS)    = ",ALPHA
460 INPUT "LIGHT ANGLE (X-Z PLANE) = ",BETA
470 INPUT "ｷｭｳﾀｲ ﾉ ｲﾛ BLUE (0-255) = ",BL
480 IF BL<0 OR BL>255 THEN 470
490 INPUT "ｷｭｳﾀｲ ﾉ ｲﾛ RED  (0-255) = ",RD
500 IF RD<0 OR RD>255 THEN 490
510 INPUT "ｷｭｳﾀｲ ﾉ ｲﾛ GREEN(0-255) = ",GR
520 IF GR<0 OR GR>255 THEN 510
530 INPUT "ﾊｲﾗｲﾄ ｷｮｳﾄﾞ (1-255)     = ",HL
540 IF HL<1 OR HL>255 THEN 530
550 LX=FNX(ALPHA,BETA)
560 LY=FNY(ALPHA)
570 LZ=FNZ(ALPHA,BETA)
600 '----------- ﾚｲ ﾄﾚｰｼﾝｸﾞ ﾃﾞｨｽﾌﾟﾚｲ -----------
610 PRW &HFF
620 FOR Y=-R1 TO R1
630   R2=SQR(R1^2-Y^2)
640   FOR X=-R2 TO R2
```

```
650      Z=SQR(R2^2-X^2)
660      CS=(X*LX+Y*LY+Z*LZ)/R1
670      IF CS<0 THEN CS=0
680      L=N :GOSUB 800:GOSUB 900:PSET (SX,SY,CL)
690      L=N1:GOSUB 800:GOSUB 900:PSET (SX+1,SY,CL)
700      IF WD=1 THEN 730
710      L=N1:GOSUB 800:GOSUB 900:PSET (SX,SY+1,CL)
720      L=N :GOSUB 800:GOSUB 900:PSET (SX+1,SY+1,CL)
730    NEXT
740 NEXT
750 PRW 0:GOTO 400
800 '----------  ﾗﾝﾀﾞﾑ ﾃﾞｨｻﾞｰ  -------------------
810 B=CS*BL+(RND(1)-.5)*M+HL*CS^4
820 R=CS*RD+(RND(1)-.5)*M+HL*CS^4
830 G=CS*GR+(RND(1)-.5)*M+HL*CS^4
840 CL=0
850 IF B>L THEN CL=CL+1
860 IF R>L THEN CL=CL+2
870 IF G>L THEN CL=CL+4
880 RETURN
900 '----------  ｽｸﾘｰﾝ ﾎﾟｼﾞｼｮﾝ  -------------------
910 SX=XO+2*X+MVX:SY=YO-WD*Y-MVY:RETURN
```

ステレオ・グラフィックス

物体を立体的に表現しようという技術は，過去において数々の研究がされてきています。原理的には，ある物体を右目と左目で眺めた場合，その物体が見える形態は，両目の幅dだけずれており，2次元の図形でも同様のずれをそれぞれの目に再現した場合には，立体的に見ることができるということになります。

ここでは，この原理を利用して，奥行のある図形，つまりリアルな図形を創出するアプリケーションを紹介します。立体視は，左右の目の距離d分だけずれた別々の透視図を作る必要があります。

ステレオ・グラフィックスの作り方

通常，平均的な視力を持っている人の場合は，目の前方約25～50㎝の所に最も強力にステレオ効果が生じるといわれています。例えば，両目の間隔を5㎝とした場合には，ステレオ角は，次の式で表すことができます。

$\theta = \tan^{-1}(5/50) = 5.710^\circ$

図5―7のように，両目の間隔をd，投影面までの距離をkとすると，もし立体視を見る人の焦点距離がk単位としたら，dは次の式で求められます。

$\tan^{-1}(d/k) = 5.71^\circ$

したがって，正しいステレオ角θ'を保つためには，次のようになります。

$d = k/10$

これにより左目の透視図は，水平に$d/1 = k/20$だけ平行移動，右側は$-d/1 = -k/20$だけ平行移動して透視投影したものであるといえます。この場合の変換マトリックスは，右目の透視図の座標値を（X_R，Y_R），左目のそれを（X_L，Y_L）とすると，次のように表現できます。

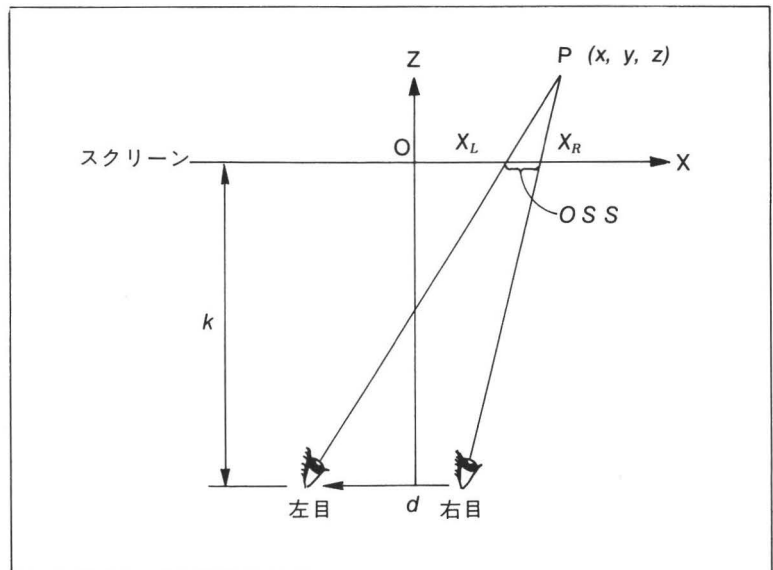



図5―7
ステレオ・グラフィックスの基礎

$$[X_R \quad Y_R \quad 1]=[x \quad y \quad 1]\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & -\frac{1}{k} \\ \frac{k}{20} & 0 & 0 & 1 \end{bmatrix} \qquad [X_L \quad Y_L \quad 1]=[x \quad y \quad 1]\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & -\frac{1}{k} \\ -\frac{k}{20} & 0 & 0 & 1 \end{bmatrix}$$

プログラムの操作方法

このプログラムをキー入力して実行（RUN）させると，第３章で作成した形状モデリングの３次元データを，ステレオめがねで立体的に見ることができます。

まず，ファイル名を聞いてくるので，第３章で作成した図形のファイル名を入力してください。

データが読み込まれると，赤と青でディスプレイ上に立体画像が描かれます。ただ，第３章で作成したデータが透視法的に画面と一致しない場合がありますので，QAX, QAY, QAZ の値を操作して，最高の位置を見いだしてください。

ステレオめがねは，右目に青，左目に赤のフィルターがあたるようにかけて物体を眺めてください。作成したデータの奥行がはっきりと現れてきます。

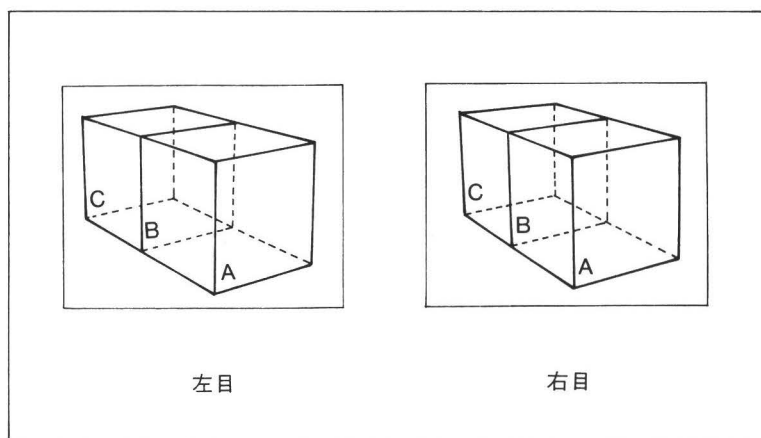



図５－８　ステレオ位置関係

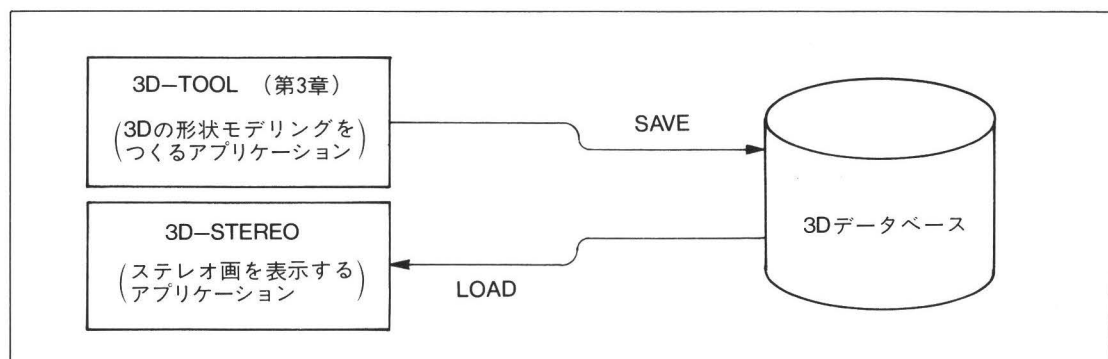



図５－９　３Ｄツールと３Ｄステレオのソフトウェア構成図

ステレオめがねの作り方

ここで紹介している「ステレオグラフィックス・プログラム」および次項の「ステレオフラクタル・プログラム」の実行画面は，ステレオめがねを通して眺めることによって，あざやかな立体像を見ることができます。ステレオめがねの作り方は，以下のとおりです。

■ステレオメガネの作り方

〔材料〕 厚手の画用紙またはボール紙（フレーム用）／赤と青のセロハン紙（5×6 cm・各1枚）

① 下の設計図を画用紙またはボール紙に写し，リンカク線に沿って長方形に切り出す。

② ヤマオリ線で折り合わせ，左右の目の部分を切り抜く。

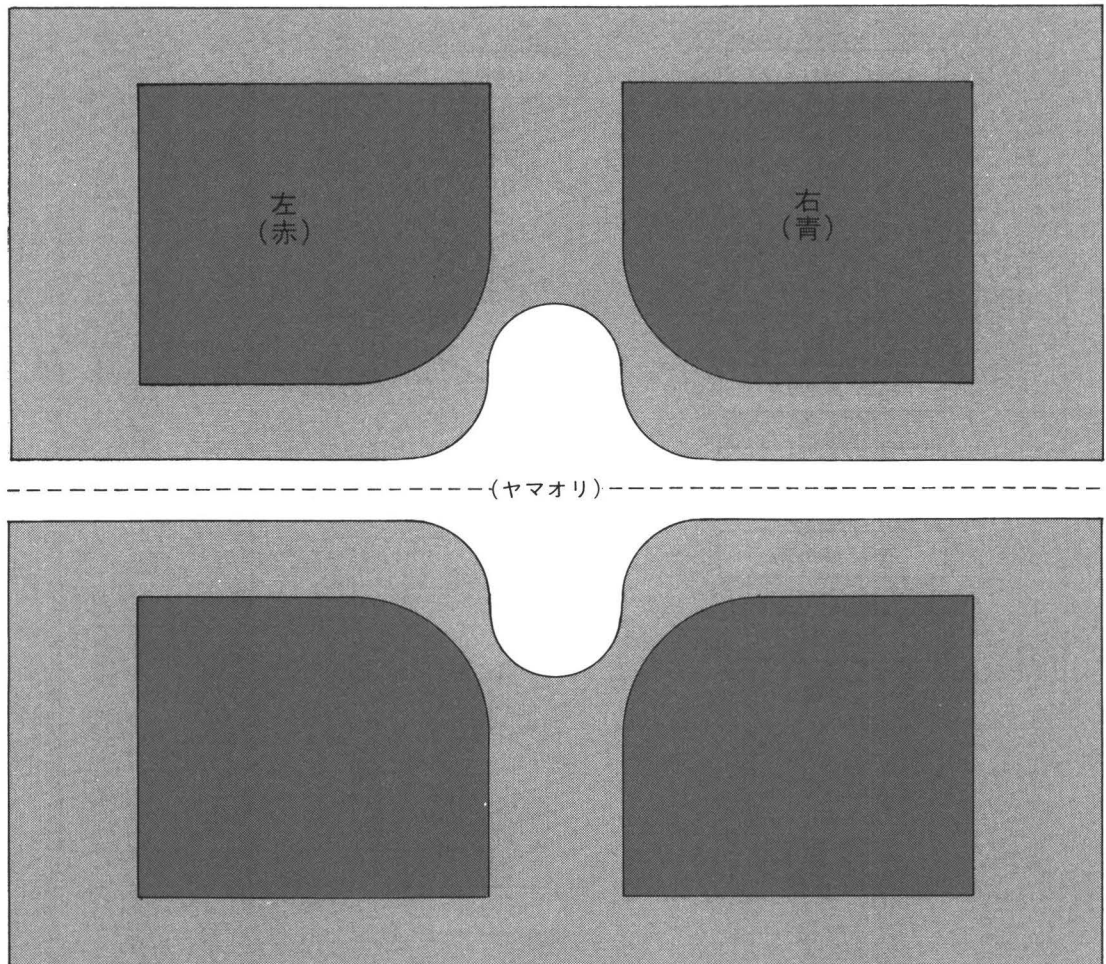
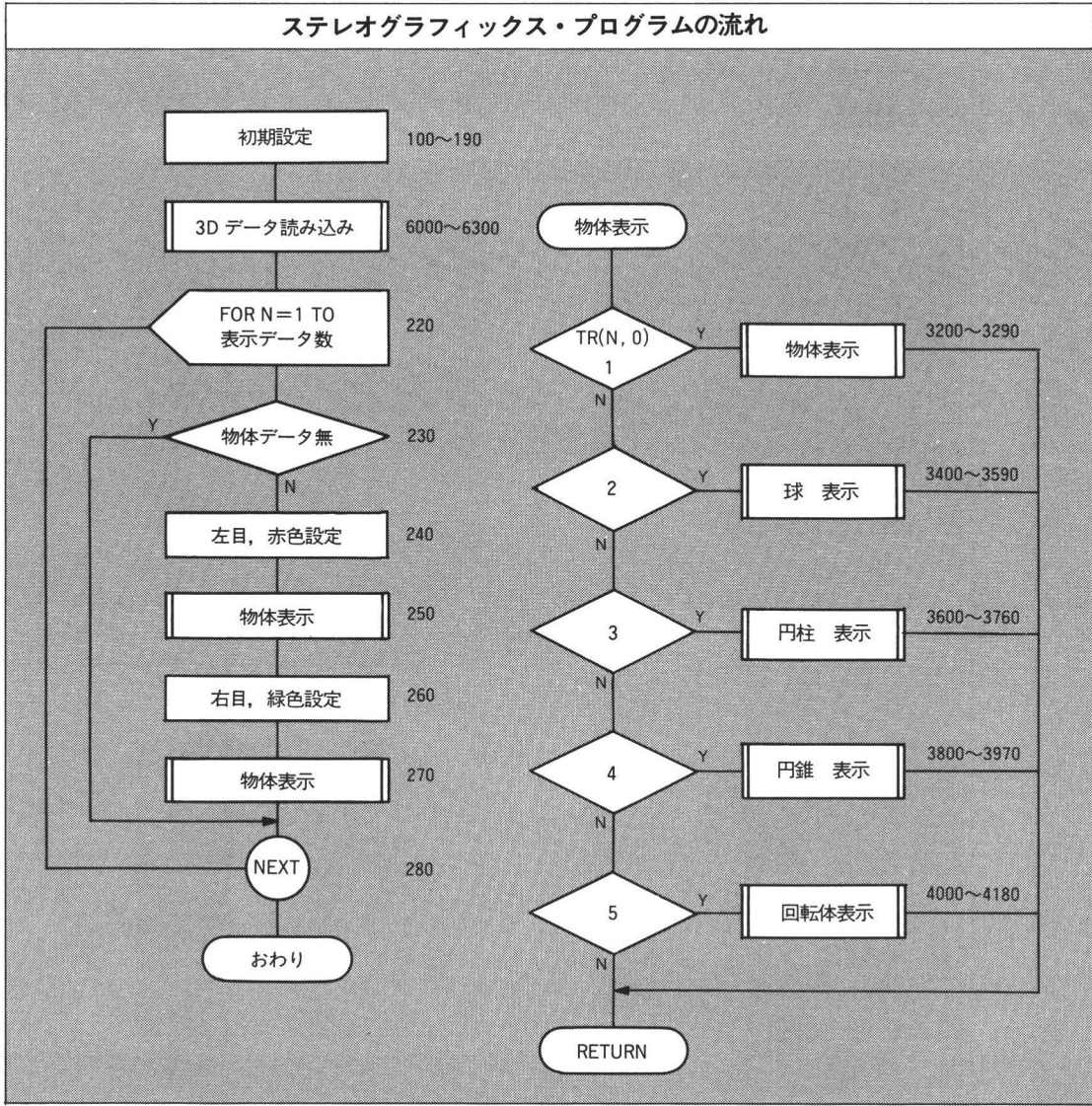
③ フレームの裏面にのりまたはペーパーボンドを塗る。

④ 赤と青のセロハン紙を片方のフレームに貼る。

⑤ ヤマオリ線を中心に貼り合わせる。

⑥ 下のリンカク線（鼻があたる部分）を切り取ってでき上がり。

■ステレオめがね設計図

（ヤマオリ）

変数とその内容			
E	左目(－1)と右目(1)の判定子	N	ラインで描く点の数
EYD	目の間隔の2分の1	PX	点の画面上X方向の位置
OFS	赤，緑で描くスクリーン上の移動量	PY	点の画面上Y方向の位置
C	ラインの色	X	データのX座標
QAX	スクリーンの移動(X方向)	Y	データのY座標
QAY	スクリーンの移動(Y方向)	Z	データのZ座標
QAZ	スクリーンの移動(Z方向)	SS	TXの正弦成分
TX	スクリーンの回転(X軸まわり)	TS	TXの余弦成分
TY	スクリーンの回転(Y軸まわり)	SF	TYの正弦成分
PSZ	目の位置(スクリーン中央からの距離)	TF	TYの余弦成分

ステレオグラフィックス・プログラムリスト

```
1 '************************************
2 '*                                  *
3 '*    3D-STEREO (WIRE FRAME) V.1.0   *
4 '*                                  *
5 '*      Programmed By [ SAIMU ]      *
6 '*                                  *
7 '*            1985 . 4 . 1           *
8 '*                                  *
9 '************************************
100 '----------- ｼｮｷｾｯﾃｲ ---------------------------
110 INIT:WIDTH 80,25,0,0:KLIST 0       :' (X1 turbo)
120 'INIT:WIDTH 80                     :' (X1       )
130 DEFINT A-N
140 APT=200:AHP=50:BHP=50:BBD=5:ABD=30:ATR=10:AR=30
150 DIM MD(BBD,ABD),HP(BHP,AHP),PT(2,APT)
160 DIM TR(ATR,13),RH(AR),RR(AR)
170 EYD=6:OFS=2:C=2
180 QAX=-50:QAY=50:QAZ=-50:STP=20:STP1=10
190 TX=-30:TY=-30:PSZ=-300:GOSUB 600
200 '----------- ﾒｲﾝ ﾙｰﾁﾝ -------------------------
210 GOSUB 6000
220 FOR N=1 TO ATR
230   IF TR(N,0)=0 THEN 280
240   E=-1:COLOR 2
250   ON TR(N,0) GOSUB 3200,3400,3600,3800,4000
260   E=-E:COLOR 1
270   ON TR(N,0) GOSUB 3200,3400,3600,3800,4000
280 NEXT:INIT:END
600 '----------- ｶｸﾄﾞ ﾉ ｹｲｻﾝ ---------------------
610 SS=SIN(RAD(TX)):SF=SIN(RAD(TY))
620 TS=COS(RAD(TX)):TF=COS(RAD(TY))
630 RETURN
3200 '----------- ﾜｲﾔｰ ﾌﾚｰﾑ ﾃﾞｨｽﾌﾟﾚｲ 1 -----------
3210 N0=TR(N,1):N1=MD(N0,0)
3220 FOR I1=1 TO N1
3230   NN1=MD(N0,I1):GOSUB 3250
3240 NEXT:RETURN
3250 I2=HP(1,NN1):GOSUB 5000:LINE (X,Y)-(X,Y),PSET
3260 FOR J=2 TO HP(0,NN1)
3270   I2=HP(J,NN1):GOSUB 5000:LINE -(X,Y)
3280 NEXT
3290 I2=HP(1,NN1):GOSUB 5000:LINE -(X,Y):RETURN
3400 '----------- ﾜｲﾔｰ ﾌﾚｰﾑ ﾃﾞｨｽﾌﾟﾚｲ 2 -----------
3410 R=TR(N,1)
3420 FOR J1=0 TO 180 STEP STP
3430   X0=R*COS(RAD(J1)):R1=R*SIN(RAD(J1))
3440   X=X0:Y=R1:Z=0
```

```
3450   GOSUB 5020:LINE (X,Y)-(X,Y),PSET
3460   FOR J2=0 TO 360 STEP STP1
3470     X=X0:Y=R1*COS(RAD(J2)):Z=R1*SIN(RAD(J2))
3480     GOSUB 5020:LINE -(X,Y)
3490   NEXT
3500 NEXT
3510 FOR J1=0 TO 180 STEP STP
3520   Y0=R*COS(RAD(J1)):R1=R*SIN(RAD(J1))
3530   X=R1:Y=Y0:Z=0
3540   GOSUB 5020:LINE(X,Y)-(X,Y),PSET
3550   FOR J2=0 TO 360 STEP STP1
3560     X=R1*COS(RAD(J2)):Y=Y0:Z=R1*SIN(RAD(J2))
3570     GOSUB 5020:LINE -(X,Y)
3580   NEXT
3590 NEXT:RETURN
3600 '-----------  ﾜｲﾔｰ ﾌﾚｰﾑ ﾃﾞｨｽﾌﾟﾚｲ 3  -----------
3610 R=TR(N,1):RH=TR(N,2)
3620 FOR ZJ1=0 TO RH STEP RH/2
3630   X=R:Y=ZJ1:Z=0
3640   GOSUB 5020:LINE(X,Y)-(X,Y),PSET
3650   FOR J2=0 TO 360 STEP STP1
3660     X=R*COS(RAD(J2)):Y=ZJ1:Z=R*SIN(RAD(J2))
3670     GOSUB 5020:LINE -(X,Y)
3680   NEXT
3690 NEXT
3700 FOR J=0 TO 360 STEP STP
3710   X=R*COS(RAD(J)):Y=RH:Z=R*SIN(RAD(J))
3720   GOSUB 5020:X1=X:Y1=Y
3730   X=R*COS(RAD(J)):Y=0:Z=R*SIN(RAD(J))
3740   GOSUB 5020:X2=X:Y2=Y
3750   LINE (X1,Y1)-(X2,Y2),PSET
3760 NEXT:RETURN
3800 '-----------  ﾜｲﾔｰ ﾌﾚｰﾑ ﾃﾞｨｽﾌﾟﾚｲ 4  -----------
3810 R1=TR(N,1):R2=TR(N,2):RH=TR(N,3)
3820 FOR ZJ1=0 TO RH STEP RH/2
3830   R=R2+(R1-R2)*(RH-ZJ1)/RH
3840   X=R:Y=ZJ1:Z=0
3850   GOSUB 5020:LINE(X,Y)-(X,Y),PSET
3860   FOR J2=0 TO 360 STEP STP1
3870     X=R*COS(RAD(J2)):Y=ZJ1:Z=R*SIN(RAD(J2))
3880     GOSUB 5020:LINE -(X,Y)
3890   NEXT
3900 NEXT
3910 FOR J=0 TO 360 STEP STP
3920   X=R1*COS(RAD(J)):Y=0:Z=R1*SIN(RAD(J))
3930   GOSUB 5020:X1=X:Y1=Y
3940   X=R2*COS(RAD(J)):Y=RH:Z=R2*SIN(RAD(J))
```

```
3950   GOSUB 5020:X2=X:Y2=Y
3960   LINE (X1,Y1)-(X2,Y2),PSET
3970 NEXT:RETURN
4000 '----------  ﾜｲﾔｰ ﾌﾚｰﾑ ﾃﾞｨｽﾌﾟﾚｲ 5  ----------
4010 FOR J1=1 TO TR(N,1):R=RR(J1)
4020   Y0=RH(J1):Y=Y0:X=R:Z=0
4030   GOSUB 5020:LINE(X,Y)-(X,Y),PSET
4040   FOR J2=0 TO 360 STEP STP1
4050     X=R*COS(RAD(J2)):Y=Y0:Z=R*SIN(RAD(J2))
4060     GOSUB 5020:LINE -(X,Y)
4070   NEXT
4080 NEXT
4090 FOR J1=0 TO 360 STEP STP
4100   X=RR(1)*COS(RAD(J1)):Y=RH(1)
4110   Z=RR(1)*SIN(RAD(J1))
4120   GOSUB 5020:LINE(X,Y)-(X,Y),PSET
4130   FOR J2=2 TO TR(N,1)
4140     X=RR(J2)*COS(RAD(J1)):Y=RH(J2)
4150     Z=RR(J2)*SIN(RAD(J1))
4160     GOSUB 5020:LINE -(X,Y)
4170   NEXT
4180 NEXT:RETURN
5000 '----------  ｶｲﾃﾝﾀｲ ﾉ ﾍﾝｶﾝ ------------------
5010 X=PT(0,I2):Y=PT(1,I2):Z=PT(2,I2)
5020 GOSUB 5500
5030 X=X-QAX:Y=Y-QAY:Z=Z-QAZ
5040 XX=X*TF+Z*SF
5050 YY=X*SS*SF+Y*TS-Z*SS*TF
5060 ZZ=-X*TS*SF+Y*SS+Z*TS*TF
5070 X=XX:Y=YY:Z=ZZ
5080 X=X-EYD*E
5090 PS=1-Z/PSZ:X=X/PS:Y=Y/PS
5100 X=320+(X+OFS*E)*2:Y=100-Y
5110 RETURN
5500 '----------  ｶｲﾃﾝﾀｲ ﾉ ﾍﾝｶﾝ ------------------
5510 K=5:GOSUB 5630:GOSUB 5620:GOSUB 5630
5520 GOSUB 5540:GOSUB 5630
5530 X=X+TX:Y=Y+TY:Z=Z+TZ:RETURN
5540 XA=X*TRY*TRZ+Y*(SRX*SRY*TRZ+TRX*SRZ)
5550 XB=Z*(TRX*SRY*TRZ-SRX*SRZ)
5560 XX=XA-XB
5570 YA=-X*TRY*SRZ-Y*(SRX*SRY*SRZ-TRX*TRZ)
5580 YB=Z*(TRX*SRY*SRZ+SRX*TRZ)
5590 YY=YA+YB
5600 ZZ=X*SRY-Y*SRX*TRY+Z*TRX*TRY
5610 X=XX:Y=YY:Z=ZZ:RETURN
5620 X=X*TX:Y=Y*TY:Z=Z*TZ:RETURN
```

```
5630 TX=TR(N,K):TY=TR(N,K+1):TZ=TR(N,K+2)
5640 IF K<>8 THEN 5690
5650 SRX=SIN(RAD(TX)):SRY=SIN(RAD(TY))
5660 SRZ=SIN(RAD(TZ))
5670 TRX=COS(RAD(TX)):TRY=COS(RAD(TY))
5680 TRZ=COS(RAD(TZ))
5690 K=K+3:RETURN
6000 '----------  ﾛｰﾄﾞ  --------------------------
6010 CLS:LOCATE 1,0:PRINT" [ LOAD ]":LOCATE 2,2
6020 INPUT"[LOAD FILE NAME ('E'= EXIT )] = ",T$
6030 IF T$="E" THEN RETURN
6040 OPEN "I",1,T$
6050 FOR I=0 TO ATR
6060   FOR J=0 TO 13
6070     INPUT #1,TR(I,J)
6080   NEXT
6090 NEXT
6100 FOR I=0 TO BBD
6110   INPUT #1,MD(I,0)
6120   FOR J=1 TO MD(I,0)
6130     INPUT #1,MD(I,J)
6140   NEXT
6150 NEXT
6160 FOR I=0 TO AHP
6170   INPUT #1,HP(0,I)
6180   FOR J=1 TO HP(0,I)
6190     INPUT #1,HP(J,I)
6200   NEXT
6210 NEXT
6220 FOR I=0 TO APT
6230   FOR J=0 TO 2
6240     INPUT #1,PT(J,I)
6250   NEXT
6260 NEXT
6270 FOR I=0 TO AR
6280   INPUT #1,RR(I),RH(I)
6290 NEXT
6300 CLOSE #1:RETURN
```

ステレオ・フラクタル

日本地図を広げ，東北地方の三陸海岸を見てください。海岸線は複雑に港や入り江で引っ込み，岬が出るという，いわゆるリアス式海岸独特の形状が見られます。もう少し詳しい東北地方だけの地図を見ても同じです。さらに詳しい地図を見ても，小さな湾や岬が複雑な形状をなしています。このように全体を見ても，その一部を見ても，同じような性質を持った曲線があります。これを，自己相同性を持つ図形といいます。

3次元空間でいうと，尾根と谷を思い浮かべてください。大きな尾根と大きな谷があるとすると，その谷の川に注ぐ小さな支流が作った谷が無数にあり，小さな尾根があります。その支流を見ても，さらに小さな川が注ぐ谷があるといった具合です。木の幹と枝，細い枝にも同様な形状が見られます。山のでこぼこも同様です。

この自己相同性という性質を取り込んで作った図形を，フラクタル図形と総称しています。

フラクタル理論の詳細な解説は，他書に譲るとして，本書では具体的にフラクタル画像を発生させるアルゴリズムについて，しかもステレオ画像作成のアルゴリズムについて解説します。

単純な平面での自己相同形（例：ヒルベルト曲線，シェルピンスキー曲線）の発生アルゴリズムは，別著「プログラミング・テクニック集」（学研版）の中でも発表していますので，ここでは3次元図形によるフラクタル画像の発生アルゴリズムについて紹介します。ステレオ法については前項の解説を参考にしてください。

フラクタル画像の作り方

フラクタル画像発生のアルゴリズムは，基本的な考え方の積み重ねの上に成り立っています。3次元フラクタル画像に入る前に，2次元図形に関する代表的な考え方から解説しましょう。

コッホ曲線による フラクタル図形の発生

単純なフラクタル図形としては，コッホ曲線があります。この発生アルゴリズムは他書でもおなじみなので，詳細は省略しますが，簡単に考え方のみを解説します。

1本の線分をAB，その中点をMとすると，MとABに対して垂線方向に移動させた場合に，AN＝NM＝MO＝OBなる点N，M，Oを求め，それらを結ぶ直線に変形させます。次いで線分AN，NM，MO，OBをそれぞれ線分ABと同じようにして4つの中点を発生させます。

このようにして，順次，細分化していきます。すると，図5—11のように，コッホ曲線を作成することができます。

中間変位による フラクタル画像の発生

次に少し複雑になりますが，3次元立体である山を，中間変位によるフラクタルで発生させるアルゴリズムについて紹介します。

三角形ABCの線分AB，BC，CAに対し，中央M_1，M_2，M_3のそれぞれの点から垂直に距離l（た

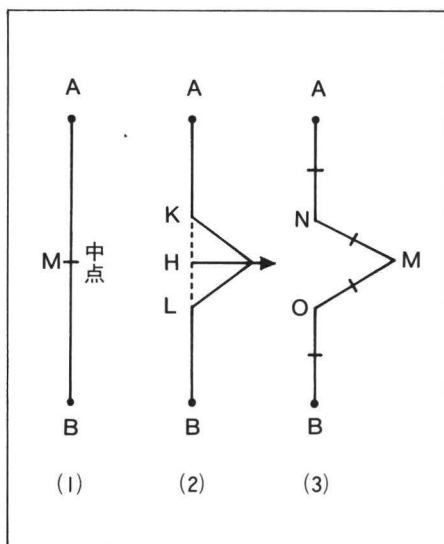



図5—10　コッホ曲線の発生

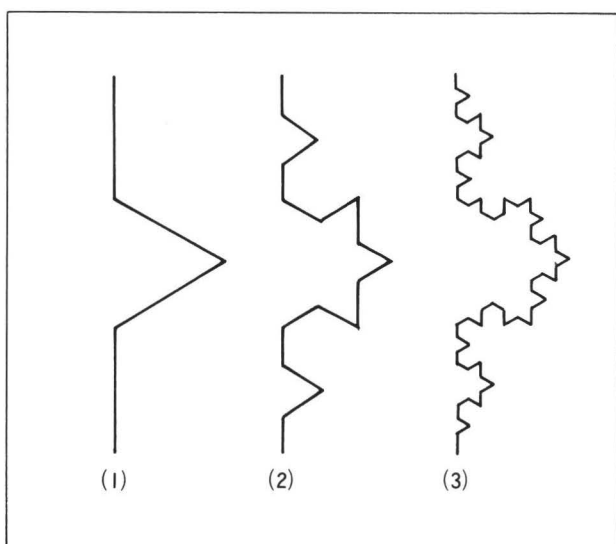



図5—11　コッホ曲線

だし *l* は変数）だけ派生させた点 L, M, N を求め，新しい線分 AL, LB, BM, MC, CN, NA, LM, MN, NL を派生させます。このようにして順次細分化していきます。

プログラムは，リストを打ち込んだ後に RUN させると，LEVEL（分割数）ごとにキー入力待ちになるので，スペースキーを押せば，6レベルまでフラクタル直線を派生させることができます（それ以上は画面をつぶしてしまうので無視します）。これを先述のステレオ・グラフィックスのアルゴリズムとマージすると，3次元の山が浮かび上がってきます。

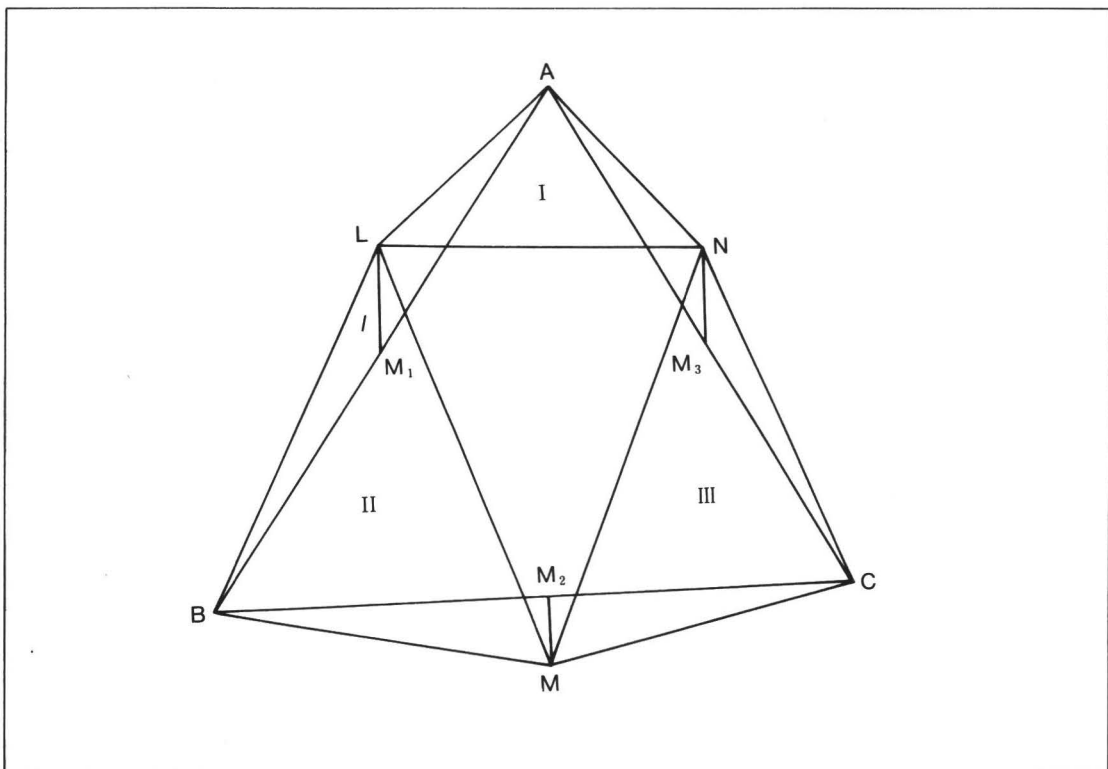
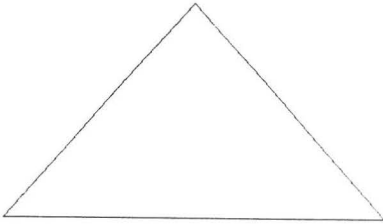
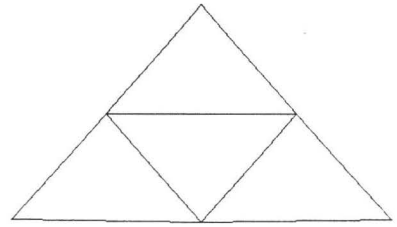
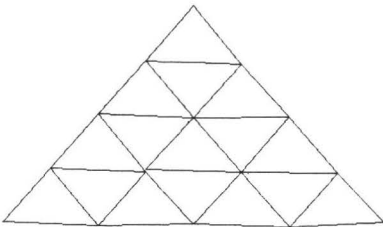
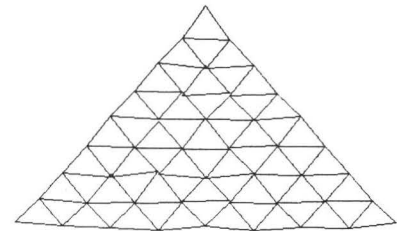



図5—12　フラクタル図形の生成

フラクタル図形の原型(ステレオ化していないもの)

変数とその内容

変数	内容	変数	内容
AX	頂点ＡのＸ座標	NX	辺CAの中点のＸ座標
AY	頂点ＡのＹ座標	NY	辺CAの中点のＹ座標
AZ	頂点ＡのＺ座標	NZ	辺CAの中点のＺ座標
BX	頂点ＢのＸ座標	E	左目(－１)と右目(１)の判定子
BY	頂点ＢのＹ座標	EYD	目の間隔の２分の１
BZ	頂点ＢのＺ座標	OFS	赤，緑で描くスクリーン上の移動量
CX	頂点ＣのＸ座標	OAX	スクリーンの移動(Ｘ方向)
CY	頂点ＣのＹ座標	OAY	スクリーンの移動(Ｙ方向)
CZ	頂点ＣのＺ座標	OAZ	スクリーンの移動(Ｚ方向)
H	深さの係数	TX	スクリーンの回転(Ｘ軸まわり)
LX	辺ABの中点のＸ座標	TY	スクリーンの回転(Ｙ軸まわり)
LY	辺ABの中点のＹ座標	PSZ	目の位置(スクリーン中央からの距離)
LZ	辺ABの中点のＺ座標	SS	TXの正弦成分
MX	辺BCの中点のＸ座標	TS	TXの余弦成分
MY	辺BCの中点のＹ座標	SF	TYの正弦成分
MZ	辺BCの中点のＺ座標	TF	TYの余弦成分

ステレオフラクタル・プログラムの流れ
はじめ
100〜130
配列の設定
140〜170
画面の設定
180
FOR NN＝1 TO 6
190
三角形頂点の
位置設定
200〜220
変数初期設定
230
フラクタル再帰
ルーチン(ステレオ)
240
キー入力待ち
250
NEXT
おわり
フラクタル再帰ルーチン
N＝N－1
310
310
N＜0
N
Y
320
D＜0
Y
N
三角形を描く
800〜 870
N＝N＋1
340
RETURN
340
中点の計算
600〜 740
STACK PUSH
1000〜1070
頂点を入れ換え
図5−12
のⅠを
描く
フラクタル再帰ルーチン
STACK PUSH
頂点を入れ換え
図5−12
のⅡを
描く
フラクタル再帰ルーチン
STACK PUSH
頂点を入れ換え
図5−12
のⅢを
描く
フラクタル再帰ルーチン
STACK PUSH
頂点を入れ換え
D＝−D
フラクタル再帰ルーチン
STACK PULL
N＝N＋1
RETURN

ステレオフラクタル・プログラムリスト

```
1 '*************************************
2 '*                                    *
3 '*         STEREO FRACTAL V.1.0       *
4 '*      file name is [FRACTAL.GKN]    *
5 '*                                    *
6 '*      Programed by    [ SAIMU ]     *
7 '*            1985 . 1.11             *
8 '*                                    *
9 '*************************************
100 '----------  ｼｮｷｾｯﾃｲ  -----------------------
110 DIM D(8),AX(8),AY(8),AZ(8),BX(8),BY(8),BZ(8)
120 DIM CX(8),CY(8),CZ(8),LX(8),LY(8),LZ(8)
130 DIM MX(8),MY(8),MZ(8),NX(8),NY(8),NZ(8)
140 INIT:OPTION SCREEN 0:KLIST 0   :'(X1 turbo     )
150 CLS 4:WIDTH 40,25,1,2          :'(X1 turbo     )
160 'INIT:OPTION SCREEN 0:KLIST 0  :'(X1 turbo M10)
170 'CLS 4:WIDTH 40,25,0,1         :'(X1 turbo M10)
180 'INIT:WIDTH 40                 :'(X1           )
190 EYD=6:OFS=2
200 QAX=-50:QAY=50:QAZ=-50
210 TX=-30:TY=-30:PSZ=-300:GOSUB 3100
220 FOR NN=1 TO 6
230   CLS4:LOCATE 0,23:PRINT "DIMENSION=";NN-1
240   AX=320:AY=40:AZ=0:BX=100:BY=160:BZ=80
250   CX=540:CY=160:CZ=40
260   P=0:N=NN-1:D=1
270   GOSUB 300
280   X$=INKEY$(0):IF X$="" THEN 240
290 NEXT:END
300 '----------  ﾌﾗｸﾀﾙ ﾘｶｰｼﾌﾞ ﾙｰﾁﾝ  --------------
310 N=N-1:IF N>=0 THEN 350
320 IF D<0 THEN D=-D:GOTO 340
330 GOSUB 800:N=N+1:RETURN
340 N=N+1:RETURN
350 GOSUB 600:GOSUB 1000
360 BX=LX:BY=LY:BZ=LZ:CX=NX:CY=NY:CZ=NZ
370 GOSUB 300
380 GOSUB 2000
390 GOSUB 1000
400 AX=LX:AY=LY:AZ=LZ:CX=MX:CY=MY:CZ=MZ
410 GOSUB 300
420 GOSUB 2000
430 GOSUB 1000
440 AX=NX:AY=NY:AZ=NZ:BX=MX:BY=MY:BZ=MZ
450 GOSUB 300
460 GOSUB 2000
470 GOSUB 1000
```

```
480 AX=MX:AY=MY:AZ=MZ:BX=NX:BY=NY:BZ=NZ
490 CX=LX:CY=LY:CZ=LZ:D=-D
500 GOSUB 300
510 GOSUB 2000
520 N=N+1
530 RETURN
600 '----------- チュウテン ノ ケイサン ------------------
610 GX=(BY-CY)*(AZ-BZ)-(AY-BY)*(BZ-CZ)
620 GY=(BX-CX)*(AZ-BZ)-(AX-BX)*(BZ-CZ)
630 GZ=(BX-CX)*(AY-BY)-(AX-BX)*(BY-CY)
640 GR=SQR(GX*GX+GY*GY+GZ*GZ)
650 KX=(AX+BX)/2:KY=(AY+BY)/2:KZ=(AZ+BZ)/2
660 GOSUB 720:LX=X9:LY=Y9:LZ=Z9
670 KX=(BX+CX)/2:KY=(BY+CY)/2:KZ=(BZ+CZ)/2
680 GOSUB 720:MX=X9:MY=Y9:MZ=Z9
690 KX=(CX+AX)/2:KY=(CY+AY)/2:KZ=(CZ+AZ)/2
700 GOSUB 720:NX=X9:NY=Y9:NZ=Z9
710 RETURN
720 HT=INT(5*RND(1))+1
730 X9=KX+HT*GX/GR:Y9=KY+HT*GY/GR:Z9=KZ+HT*GZ/GR
740 RETURN
800 '----------- サンカッケイ ヲ エガ゛ク ------------------
810 E=-1:PC=2:GOSUB 820:PC=1:E=-E
820 X=AX:Y=2*AY:Z=AZ            :'(X1 turbo          )
830 'X=AX:Y=AY:Z=AZ             :'(X1 turbo M10 & X1)
840 GOSUB 3000
850 X1=PX:Y1=PY
860 X=BX:Y=2*BY:Z=BZ            :'(X1 turbo)
870 'X=BX:Y=BY:Z=BZ             :'(X1 turbo M10 & X1)
880 GOSUB 3000
890 X2=PX:Y2=PY
900 X=CX:Y=2*CY:Z=CZ            :'(X1 turbo          )
910 'X=CX:Y=CY:Z=CZ             :'(X1 turbo M10 & X1)
920 GOSUB 3000
930 X3=PX:Y3=PY
940 LINE (X1,Y1)-(X2,Y2),PSET,PC
950 LINE (X2,Y2)-(X3,Y3),PSET,PC
960 LINE (X3,Y3)-(X1,Y1),PSET,PC
970 RETURN
1000 '----------- ヘンスウ ヲ タイヒ スル -----------------
1010 P=P+1:AX(P)=AX:AY(P)=AY:AZ(P)=AZ
1020 BX(P)=BX:BY(P)=BY:BZ(P)=BZ
1030 CX(P)=CX:CY(P)=CY:CZ(P)=CZ
1040 LX(P)=LX:LY(P)=LY:LZ(P)=LZ
1050 MX(P)=MX:MY(P)=MY:MZ(P)=MZ
1060 NX(P)=NX:NY(P)=NY:NZ(P)=NZ
1070 D(P)=D:RETURN
```

```
2000 '----------  ヘンスウ ヲ モト゛ス  -----------------
2010 AX=AX(P):AY=AY(P):AZ=AZ(P)
2020 BX=BX(P):BY=BY(P):BZ=BZ(P)
2030 CX=CX(P):CY=CY(P):CZ=CZ(P)
2040 LX=LX(P):LY=LY(P):LZ=LZ(P)
2050 MX=MX(P):MY=MY(P):MZ=MZ(P)
2060 NX=NX(P):NY=NY(P):NZ=NZ(P)
2070 D=D(P):P=P-1:RETURN
3000 '----------  TRANS ルーチン  -----------------
3010 X=X-QAX:Y=Y-QAY:Z=Z-QAZ
3020 XX=X*TF+Z*SF
3030 YY=X*SS*SF+Y*TS-Z*SS*TF
3040 ZZ=-X*TS*SF+Y*SS+Z*TS*TF
3050 X=XX:Y=YY:Z=ZZ
3060 X=X-EYD*E
3070 PS=1-Z/PSZ:X=X/PS:Y=Y/PS
3080 PX=X+OFS*E:PY=Y
3090 RETURN
3100 '----------  カクト゛ ノ ケイサン  -----------------
3110 SS=SIN(RAD(TX)):SF=SIN(RAD(TY))
3120 TS=COS(RAD(TX)):TF=COS(RAD(TY))
3130 RETURN
```

付録1

画面のSAVE/LOADプログラム

作成したグラフィックスを，フロッピーにSAVE／LOADするためのサブルーチンを紹介します。本書のすべてのプログラムで作成したグラフィックスについて使用可能です。基本的には，どのプログラムにも，SAVE／LOADルーチンが付いていますので，作成した画面を読み出す場合にお使いください。

プログラムを実行（RUN）すると，画面上に次のように表示されます。

```
1. LOAD
2. SAVE
3. END
```

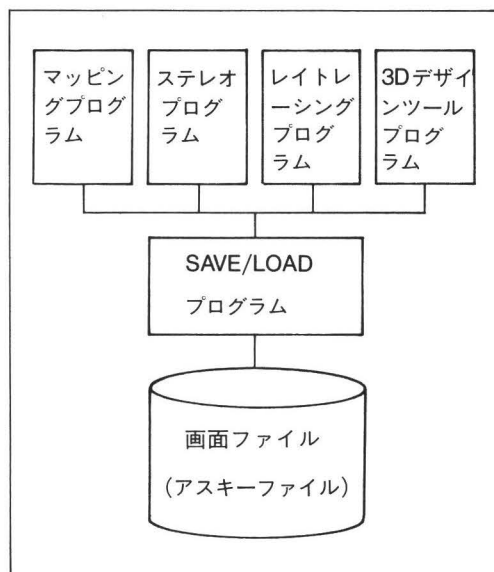
画面のSAVE／LOADプログラム

```
1 '************************************
2 '*                                  *
3 '*          SAVE & LOAD             *
4 '*                                  *
5 '*     Programmed By [ SAIMU ]      *
6 '*                                  *
7 '*          1985 . 4 . 1            *
8 '*                                  *
9 '************************************
100 '----------- セーフ゛ & ロート゛ ----------------------
110 INIT:CLS:KLIST 0
115 FOR I=0 TO 7:PAL(I)=I:NEXT
120 PRINT " 1. LOAD ":PRINT " 2. SAVE ":PRINT " 3. END "
130 PRINT "  Push Key 1 OR 2 OR 3 !!"
140 K$=INKEY$(1):IF K$="" THEN 140
150 IF K$="1" THEN GOSUB 200:GOTO 110
160 IF K$="2" THEN GOSUB 400:GOTO 110
170 IF K$="3" THEN GOTO 190
180 GOTO 140
190 CLS:INIT:END
200 '----------- カ゛メン ノ ロート゛ ----------------------
210 CLS:PRINT"[ LOAD ]"
220 INPUT"  FILE NAME ? ('/'=RETURN) ",T$
230 IF LEN(T$)>16 THEN 200
240 IF T$="/" THEN RETURN
250 GOSUB 290
260 FOR I=0 TO 7
270   PALET I,PAL(I)
280 NEXT:RETURN
290 OPTIONSCREEN 4:PRW &HFF
300 OPEN"I",1,T$:REC=0
```

作成した画面を呼び出したいときは1，SAVEしたいときは2，プログラムを終了したい場合は3を選んでください。SAVE／LOADのファイル名は，ファイルディスクリプタを指定した後，13文字以内で指定してください。

当プログラムでは，5インチのディスクに，6枚の画像をファイルすることが可能です。なお，同一名でファイルすると，前に作成した絵の上に画面がSAVEされますので注意してください(作成した画面の安全を守るために，ロックをかけることをおすすめします)。

ロックは，次のようにしてかけます。

SET "ファイルネーム"，"P"

```
310 GOSUB 610:IF WI<>80 THEN RETURN 200
320 A$=INPUT$(128,1):B$=INPUT$(128,1)
330 DEVO$"MEM:",REC,A$,B$:REC=REC+1
340 IF REC<&HC0 THEN 320
350 CLOSE #1:PRW 0:INIT
360 OPTIONSCREEN 0:CLS:RETURN
400 '----------  ｶﾞﾒﾝ ﾉ ｾｰﾌﾞ  ---------------------
410 PRW 0:CLS
420 PRINT"[ SAVE ]"
430 INPUT"  FILE NAME ? ('/'=RETURN) ",T$
440 IF LEN(T$)>16 THEN GOTO 400
450 IF T$="/" THEN RETURN
460 OPTIONSCREEN 4:PRW &HFF
470 OPEN"O",1,T$:REC=0
480 PRINT #1,80
490 FOR I=0 TO 7
500   PRINT #1,PAL(I)
510 NEXT
520 DEVI$"MEM:",REC,A$,B$:PRINT#1,A$;B$;
530 REC=REC+1
540 IF REC<&HC0 THEN 520
550 GOTO 350
600 '----------  WIDTH ﾁｪｯｸ  ---------------------
610 INPUT #1,WI
620 IF WI<>80 THEN 660
630 FOR I=0 TO 7
640   INPUT #1,PAL(I)
650 NEXT:RETURN
660 GOSUB 350
670 PRINT"WIDTH IS NOT 80 !! (Push Any key) ";
680 KY$=INKEY$(1)
690 RETURN
```

付録2

マトリックス演算の基礎知識

本書で解説している演算式は，高等学校の数学IIBで学ぶ行列の演算（マトリックス演算）を使用しています。ここでその基礎的内容についてまとめておきます。

1．マトリックス演算の用語

マトリックスを構成する数の列をマトリックスの要素といいます。マトリックスは，これらの要素により成り立っており，行と列があります。行は横方向をさし，列は縦方向をさしていますが，行の数と列の数が等しい場合，そのマトリックスは正方であるといわれます。ここでCGの変換式で利用した4×4正方マトリックスを例にとって考えると，次のように書き表すことができます。

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix}$$

ここで最初の添字は，マトリックスの行，2つ目の添字はマトリックスの列を示しています。例えば a_{43} は，第4行，第3列の要素であるといえます。一般的にm行n列のマトリックスは，次数 $m \times n$のマトリックスといえます。

上記4×4マトリックスでは，$m=n$であり，a_{11}, a_{22}, a_{33}, a_{44} のような要素を対角要素と呼んでいます。これらの対角要素の和はトレースと呼ばれます。

単位マトリックスとは，この対角要素がすべて1で，その他の要素が0のマトリックスをいいます。先ほどの例でいうと，

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

となれば4×4の単位マトリックスといえます。

マトリックス演算においては，1つのマトリックスの各要素が，他のマトリックスの対応する要素と等しい場合においてのみ，その2つのマトリックスは互いに等しいといえます。

2．マトリックスの和と差

2つのマトリックスが同一次数のときに，マトリックスの和と差が定義できます。例をあげて解説します。

マトリックスの和は，それぞれの対応する各要素の和を求めればよく，次のように計算できます。

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix} + \begin{bmatrix} b_{11} & b_{12} & b_{13} & b_{14} \\ b_{21} & b_{22} & b_{23} & b_{24} \\ b_{31} & b_{32} & b_{33} & b_{34} \\ b_{41} & b_{42} & b_{43} & b_{44} \end{bmatrix}$$

$$= \begin{bmatrix} a_{11}+b_{11} & a_{12}+b_{12} & a_{13}+b_{13} & a_{14}+b_{14} \\ a_{21}+b_{21} & a_{22}+b_{22} & a_{23}+b_{23} & a_{24}+b_{24} \\ a_{31}+b_{31} & a_{32}+b_{32} & a_{33}+b_{33} & a_{34}+b_{34} \\ a_{41}+b_{41} & a_{42}+b_{42} & a_{43}+b_{43} & a_{44}+b_{44} \end{bmatrix}$$

同様に，マトリックスの差は，次のように計算できます。

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix} - \begin{bmatrix} b_{11} & b_{12} & b_{13} & b_{14} \\ b_{21} & b_{22} & b_{23} & b_{24} \\ b_{31} & b_{32} & b_{33} & b_{34} \\ b_{41} & b_{42} & b_{43} & b_{44} \end{bmatrix}$$

$$= \begin{bmatrix} a_{11}-b_{11} & a_{12}-b_{12} & a_{13}-b_{13} & a_{14}-b_{14} \\ a_{21}-b_{21} & a_{22}-b_{22} & a_{23}-b_{23} & a_{24}-b_{24} \\ a_{31}-b_{31} & a_{32}-b_{32} & a_{33}-b_{33} & a_{34}-b_{34} \\ a_{41}-b_{41} & a_{42}-b_{42} & a_{43}-b_{43} & a_{44}-b_{44} \end{bmatrix}$$

3. マトリックス演算の積

マトリックス演算の積は，CGにとって最も大切な計算式です。次数が $n_1 \times m_1$ と $n_2 \times m_2$ の2つのマトリックスの積を定義するためには，最初のマトリックスの列数，つまり m_1 と，2つ目のマトリックスの行数，つまり n_2 の数が一致していなければなりません。これが演算で行ううえでの条件です。例に従って計算してみましょう。

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \\ a_{41} & a_{42} & a_{43} \end{bmatrix} \begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{bmatrix}$$

$$= \begin{bmatrix} a_{11}b_{11}+a_{12}b_{21}+a_{13}b_{31} & a_{11}b_{12}+a_{12}b_{22}+a_{13}b_{32} & a_{11}b_{13}+a_{12}b_{23}+a_{13}b_{33} \\ a_{21}b_{11}+a_{22}b_{21}+a_{23}b_{31} & a_{21}b_{12}+a_{22}b_{22}+a_{23}b_{32} & a_{21}b_{13}+a_{22}b_{23}+a_{23}b_{33} \\ a_{31}b_{11}+a_{32}b_{21}+a_{33}b_{31} & a_{31}b_{12}+a_{32}b_{22}+a_{33}b_{32} & a_{31}b_{13}+a_{32}b_{23}+a_{33}b_{33} \\ a_{41}b_{11}+a_{42}b_{21}+a_{43}b_{31} & a_{41}b_{12}+a_{42}b_{22}+a_{43}b_{32} & a_{41}b_{13}+a_{42}b_{23}+a_{43}b_{33} \end{bmatrix}$$

この例でも明らかなように，一般的に $n_1 \times m_1$ のマトリックスと $n_2 \times m_2$ のマトリックスの積をとると，$n_1 \times m_2$のマトリックスとなることがわかります。

ここでマトリックスの乗算では，次の関係式が成り立ちます。

●可換ではない。

$[A][B] \neq [B][A]$

●第1次分配則が成り立つ

$A(B+C)=AB+AC$

●第2分配則が成り立つ

$(A+B)C=AC+BC$

●結合則が成り立つ

$A(BC)=(AB)C$

4.正方マトリックスの行列計算方法

正方マトリックスの行列式は，$|A|$で表現されます。3×3のマトリックスをもってこれを解くと，

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$$

の行列は，

$$\begin{vmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{vmatrix} = a_{11}\begin{vmatrix} a_{22} & a_{23} \\ a_{32} & a_{33} \end{vmatrix} - a_{12}\begin{vmatrix} a_{21} & a_{23} \\ a_{31} & a_{33} \end{vmatrix} + a_{13}\begin{vmatrix} a_{21} & a_{22} \\ a_{31} & a_{32} \end{vmatrix}$$

と表現できます。これを解くと，次のようになります。

$$\begin{vmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{vmatrix} = a_{11}(a_{22}a_{33}-a_{23}a_{32})-a_{12}(a_{21}a_{33}-a_{23}a_{31})+a_{13}(a_{21}a_{32}-a_{22}a_{31})$$

5.正方マトリックスの逆マトリックス

正方マトリックスにおいて逆マトリックスを求めるには，次の関係式が成り立ちます。

$[A][X]=[Y]$　　ならば　$[X]=[A]^{-1}[Y]$

ここで〔A〕$^{-1}$は，正方マトリックス〔A〕の逆マトリックスです（ここでマトリックスの行列が0でないならば，ただ1つだけこの逆マトリックスが存在します）。

次に，逆マトリックスの重要な性質について述べます。

任意のマトリックス〔A〕に対して，

$[A][A]^{-1}=[I]$　　ただし　[I]は単位マトリックス

の等式が成り立ちます。

付録3

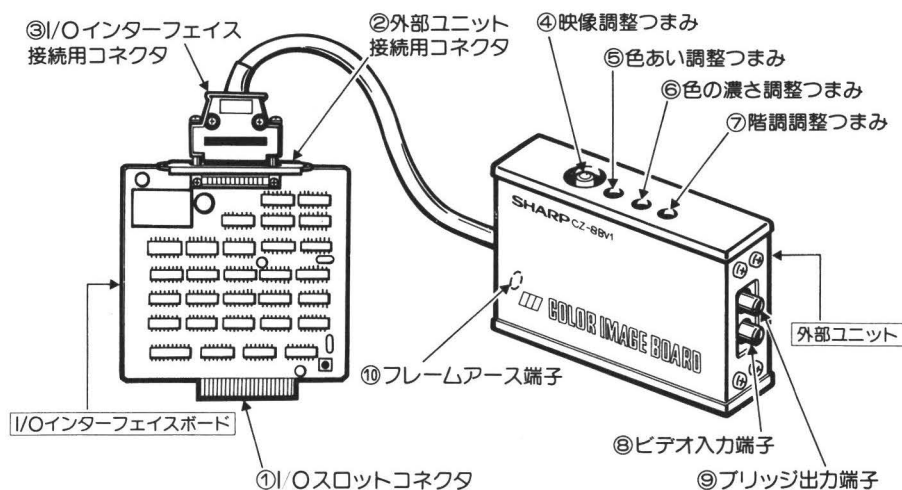
イメージボードを使った自動画像入力

第2章で紹介した2次元画像処理用のデザインツールを使えば，読者諸氏がそれぞれイメージされる独自の画像を入力することが可能です。ところで，いちいち絵を描いていくのはどうも面倒だ，自動的に取り込めるハードはないものかといった人のために，新しい入力装置が開発されました。

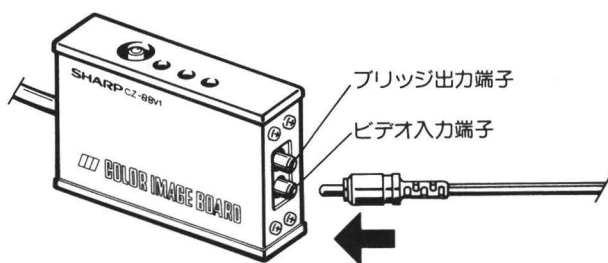
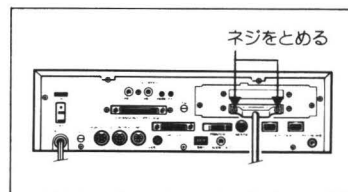
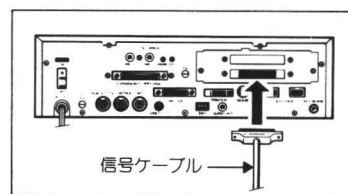
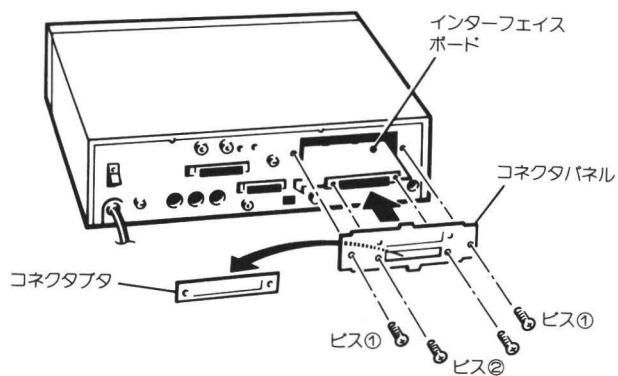
カラーイメージボード CZ-8BV1がそれです。このボードをX1ターボ／X1シリーズに装着すると，テレビ，ビデオ，ビデオカメラ，ビデオディスク等の映像を，カラー静止画像としてパソコンテレビX1ターボ／X1シリーズにそのまま入力することができます。

取り込んだ画像は，拡大／縮小／切り抜き／輪郭抽出等，自在に修正加工ができ，カセットテープやフロッピーディスクに保存したり再生したりすることや，プリンタでハードコピーをとることも可能です。また，コマ落としや4画面／16画面のマルチストロボアクション効果の演出が可能です。

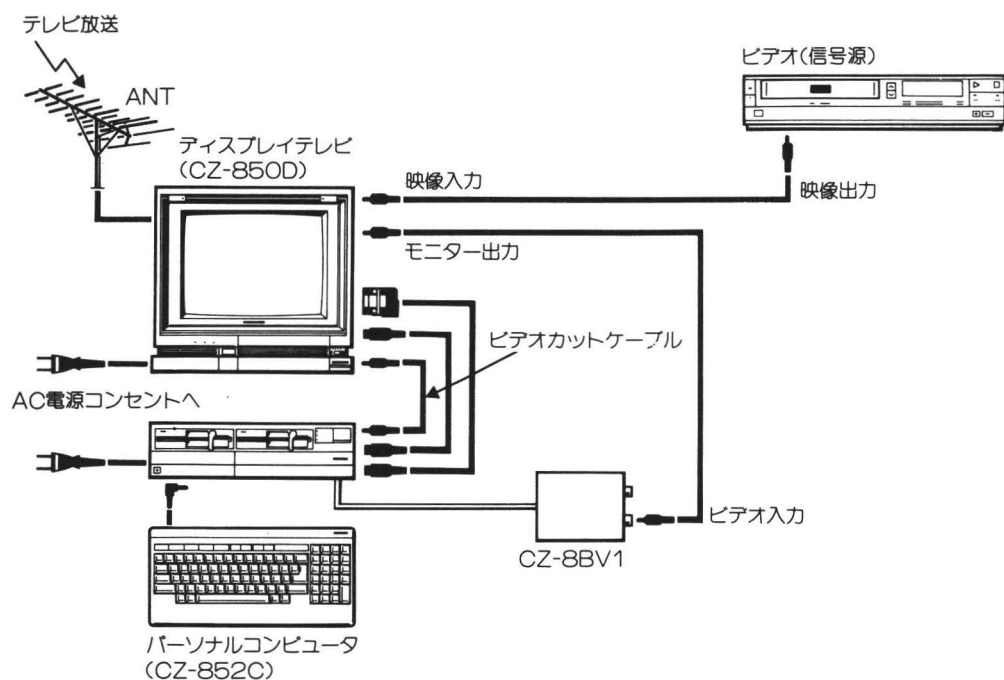
1．カラーイメージボードの本体への装着

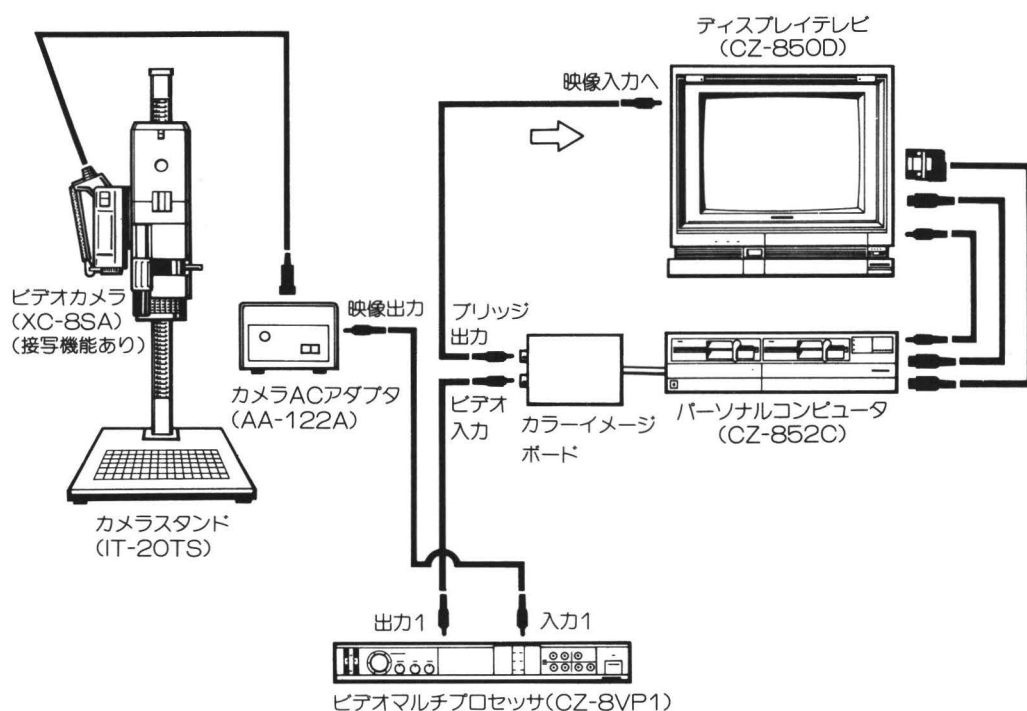
カラーイメージボード CZ-8BV1を購入すると，下図のようなハードが梱包されています。

まず，I／Oインターフェイスボードをコンピュータの拡張I／Oスロットに接続，次に，外部ユニットのケーブルコネクタをI／Oインターフェイスボードのコネクタに接続します（コネクタは両端のネジをドライバーでしっかりとめる）。次にコンピュータ本体と外部ユニットのそれぞれのフレームアース端子（FG）を付属のブライドワイヤーで接続（雑音を多く発生する機器が近くにある場合に，カラーイメージボードからの画像を安定させるために有効）します。

最後に，外部ユニットのビデオ入力端子に入力したい機器の出力端子（例えば，テレビ放送を直接入力したいときはテレビからのモニター出力を，VTRからの映像を直接入力したい場合はVTRの映像出力）を接続します。外部ユニットのブリッジ出力端子は，ビデオ入力端子に入れた映像を別のディスプレイ等にモニターする場合に使用します。

インターフェイス
ボード
コネクタパネル
コネクタブタ
ビス①
ビス①
ビス②
信号ケーブル
ネジをとめる
SHARP CZ-8BV1
COLOR IMAGE BOARD
ブリッジ出力端子
ビデオ入力端子

2．外部ユニットとの接続例

(1) テレビ放送と VTR からの映像を得る接続法

VTR にカメラを接続すると，カメラからの情報も入力することができます。

(2) **ビデオカメラとビデオマルチプロセッサを使って映像を得る接続法**

ビデオカメラからの画像情報を色補正する場合は，ビデオマルチプロセッサ（CZ-8VP1）を併用すると，種々の色補正ができて便利です。

これらの接続例をもとに，「カラーイメージツール」のソフトウェアを使って，各種入力装置からのイメージ画面を取り込んでください。なお，取り込んだ画像は，付録1（176ページ）の SAVE／LOAD プログラムを起動させて画面を SAVE した後，種々のプログラムで処理することができます。例えば，本書で紹介したマッピング・プログラムを使えば，種々の画像変換が可能になります。

カラーイメージボードを下記の機器で使用する場合には，オプションとして表中の○印のハードが必要ですのでご注意ください。

オプション／機種	グラフィックRAMボード		拡張I/Oポート	拡張I/Oボックス
	CZ-8GR	CZ-8BGR2	CZ-8EP	CZ-81EB(S/R)
CZ-800C	○	—	○	—
CZ-801C	—	—	—	○
CZ-802C	—	—	○	—
CZ-850C	—	○	—	—

■著者プロフィール

畠中兼司（Kenji Hatakenaka）

1947年	和歌山県に生まれる
1971年	金沢美術工芸大学美術工芸学部卒業
1973年	京都工芸繊維大学大学院工芸学研究科修了
1973～79年	4月　ソニー株式会社
1980年	シャープ株式会社入社 同社総合デザイン本部にてデザイン企画業務およびCADシステム開発業務に従事し、現在に至る。
現　在	日本デザイン学会会員 情報処理学会会員 意匠学会会員 NICOGRAPH企画委員 JAGDAコンピュータ部会会員 兵庫県地域活性化委員会委員 大阪デザイナー専門学校特別講師 グループ「彩夢」代表

主な著書および発表論文

1983年	「パソコン・グラフィックスの作り方楽しみ方」(学研)
1984年	コンピュータ・グラフィックス東京'84「A PRACTICAL APPLICATION OF A COMPUTER TO INDUSTRIAL DESIGN」論文発表
1984年	読売新聞関西版特集記事「情報新時代」に、コンピュータ・グラフィックス連載 「プログラミングテクニック集」(学研)
1985年	「特選グラフィックス・デザイン」(共著)(学研) コンピュータ・グラフィックス東京'85「PRODUCTION OF ARTISTIC IMAGES WITH "ART PROCESSOR"」論文発表

X1ターボ／X1シリーズで楽しむ
3次元グラフィックスの描き方

166 745

畠中兼司著

発行人――――児山敬一

編集人――――坪田良彦

発行所――――株式会社　学習研究社

東京都大田区上池台4－40－5

郵便番号145 振替口座 東京8－142930

電話 東京03(726)8111（受付案内台）／03(464)3951（編集部）

印刷所――――中央精版株式会社／株式会社廣済堂

●この本の内容・製品に関するお問い合わせがありましたら下記あてにお願いします。

文書は（〒145）東京都大田区上池台4－40－5

学研お客さま相談センター「X1　3次元グラフィックス」係

電話は　東京03(726)8124

X1ターボ・X1シリーズ **3次元グラフィックスの描き方**

掲載プログラム パック サービスのお知らせ

◇ ◇ ◇

このたびは「X1ターボ・X1シリーズ3次元グラフィックス」をお買いあげいただき，ありがとうございました。本書によりBASICによる3次元図形処理の手法についてご理解を深めていただけましたら幸いでございます。

本書掲載のプログラム中より，下記のプログラムを収録したフロッピーディスク（5インチ，両面倍密）をおわけしております。ご希望のかたは，下記の要領にてお申し込みください。

収録プログラムとサンプルデータ

- **デザインツールプログラム**
  カラー平面画の製作ツールです。
- **3次元デザインツールプログラム**
  ワイヤーフレームモデルの製作ツールです。
- **各種マッピングプログラム**
  球体マッピングプログラム
  円柱マッピングプログラム
  円錐マッピングプログラム
  直方体マッピングプログラム
  回転体マッピングプログラム
  （マッピング機能をもっています。）
- **レイトレーシングプログラム**
  球体のレイトレースができます。
- **ステレオグラフスックスプログラム**
  ワイヤーフレーム・フラクタル
- **画面セーブロードプログラム**
- その他，**画像サンプルデータファイル**

お申し込み方法

①申し込み先

〒150 東京都渋谷区道玄坂2-10-7
新大宗ビル2号館15階
学研コンピュータ出版「3次元グラフィックス」係

②価格

Ⓐ5インチフロッピーディスク版(2まい 7400円)
上記価格には荷造送料を含みません。荷造送料として600円のご負担をお願いいたします。

③送金方法

Ⓐ下記申し込み書に必要事項をご記入のうえ，前金でお申し込みください。(現金書留か郵便小為替に限ります。)

Ⓑお申し込み書到着後，10日以内に発送する予定です。

④その他

受領した前金は，事情のいかんを問わず一切返金いたしません。ただし，リードエラーがあった場合は現品と引き換えに代品をお送りします。

------------------------------ 切りとり線 ------------------------------

||||||||| 申し込み書 |||||||||

「X1ターボ・X1シリーズ**3次元グラフィックの描き方**」プログラムパックを申し込みます。

申し込み者	ご住所	〒□□□-□□					
	お名前	（ 歳）	㊞	電話番号 （ ）			
	申し込み者が未成年(20歳未満)の場合は保護者の同意が必要です	保護者お名前 ㊞					
ご使用機種(○をおつけください)		X1, X1cs, X1Dc, X1ck, X1ターボタイプ( ), X1ターボ2					
ご連絡欄	数量		金額	円	荷造送料	600円	
					計	円	

ご住所・お名前は楷書でていねいにお書きください。アパート・マンションにお住まいの方はアパート名，部屋番号までご記入ください。 A